Broad-Coverage Rule-Based Processing of Temporal Expressions

Paweł Piotr Mazur
Master of Science (M.Sc.)

MACQUARIE UNIVERSITY
Centre for Language Technology
Department of Computing
Faculty of Science

and

WROCŁAW UNIVERSITY OF TECHNOLOGY
Software Engineering Department
Institute of Informatics
Faculty of Computer Science and Management

This thesis is presented for the degree of
DOCTOR OF PHILOSOPHY

Submitted in partial fulfilment of joint institutional requirements
for the double-badged degree

December 2011

Contents

List of Tables

List of Figures

Abstract

This thesis presents work concerning the processing of **temporal expressions** in text documents, addressing both the identification of such expressions in texts (the **recognition** stage) and the derivation of their meaning (the **interpretation** stage). In scientific literature, the term 'temporal expression' has been used very broadly to denote different things. In our work this term is used to refer to sequences of words which carry information about either *when* or *how often* things happen, or *how long* they last.

The ultimate aim of this work is to advance our ability to implement robust broad-coverage systems that can be applied to collections of documents to extract information about time. This is motivated by the important role that is played by temporal information, reflected both in how often we refer to time in everyday communication and by the large number of natural language applications that require a precise understanding of the information expressed in temporal expressions: correct identification and interpretation of these expressions in texts is one of the key elements in many NLP tasks such as information extraction, question answering, text summarization and the temporal indexing of documents.

Taking the view that it is necessary to carry out a systematic analysis and categorization of temporal expressions in order to successfully design and implement a robust, large-scale, broad-coverage computer system that is capable of extracting temporal information (a temporal expression tagger), we start with the construction of a **taxonomy** of temporal expressions.

In our approach to expressing the semantics of temporal expressions we draw a clear distinction between what we call the **local** and the **global** semantics of these expressions. The local semantics is the meaning of an expression without any context: it is purely the combined meaning of the lexical items of which the expression is built. The global semantics represents the value of the expression in the context of the whole document. Following this distinction we propose a set of extensions to existing annotation schemes; called **LTIMEX**, these extensions provide a level of data annotation that supports both increased modularity in tagger design and more detailed evaluation of taggers.

In the thesis we present a number of experiments concerned with solving specific problems related to the recognition and interpretation of temporal expressions. This includes syntax-based extent recognition, the interpretation of references to bare weekday names, and finding the reference time for the interpretation of context-dependent expressions.

The practical part of the thesis deals with the technical aspects of constructing a temporal expression tagger; in particular it presents our robust and broad-coverage tagger called **DANTE**, which achieves state-of-the-art accuracy and performance. We make this system publicly available to foster further research involving the extraction of temporal information from text. Another resource which we have developed and describe in the thesis, and which we also make publicly available, is the **WikiWars** corpus; a TIMEX2-annotated dataset containing narratives sourced from Wikipedia articles about military conflicts. The domain and text genre involved make this a unique resource.

Streszczenie

Niniejsza rozprawa przedstawia badania dotyczące przetwarzania **wyrażeń temporalnych** w dokumentach tekstowych, zarówno na poziomie identyfikacji takich wyrażeń w tekście (**rozpoznawanie**) jak i odkrywania ich znaczenia (**interpretacja**). W literaturze naukowej pojęcie 'wyrażenie temporalne' było używane w dość luźny sposób w kontekście różnych zagadnień. W niniejszej pracy pojęcie to używane jest w odniesieniu do ciągów słów, które wyrażają informację o tym *kiedy* lub *jak często* rzeczy się zdarzają, lub *jak długo* trwają.

Głównym celem niniejszej pracy jest udoskonalenie naszych zdolności w zakresie konstruowania niezawodnych systemów komputerowych do przetwarzania kolekcji dokumentów w celu wydobywania zawartych w nich informacji dotyczących czasu. Cel ten umotywowany jest znaczącą rolą informacji temporalnej; ważność ta przejawia się zarówno tym, jak często odnosimy się do czasu w codziennej komunikacji, jak i dużą liczbą zastosowań z dziedziny przetwarzania języka naturalnego, które wymagają dokładnego rozumienia informacji zawartej w wyrażeniach temporalnych: poprawne rozpoznawanie i interpretacja tych wyrażeń jest jednym z kluczowych elementów w wielu zadaniach NLP takich jak wydobywanie informacji, udzielanie odpowiedzi, streszczanie tekstu, czy temporalne indeksowanie dokumentów.

W pracy przyjęto, że w celu zaprojektowania i implementacji niezawodnego i skalowalnego systemu przetwarzającego wyrażenia temporalne, konieczne jest przeprowadzenie systematycznej analizy i kategoryzacji tych wyrażeń. W związku z tym, pierwszym krokiem było opracowanie **taksonomii** wyrażeń temporalnych.

W podejściu do zagadnienia semantyki wyrażeń temporalnych przyjęto wyraźne rozróżnienie pomiędzy semantyką **lokalną** i **globalną**. Semantyka lokalna dotyczy znaczenia wyrażenia temporalnego bez uwzględnienia jakiegokolwiek kontekstu użycia: na tym poziomie korzysta się jedynie ze znaczenia jednostek leksykalnych składających się na wyrażenie. Semantyka globalna reprezentuje wartość wyrażenia w kontekście całego dokumentu. W oparciu o to rozróżnienie przedstawiono zestaw rozszerzeń do istniejących schematów oznaczania wyrażeń temporalnych. Opracowane rozwiązanie, nazwane **LTIMEX**, wprowadza nowy poziom anotacji informacji temporalnej, który pomaga zapewnić większą modułowość systemów oznaczających wyrażenia temporalne oraz jest przydatny w przeprowadzaniu dokładniejszej oceny i analizy jakości tych systemów.

W rozprawie przedstawiono szereg eksperymentów dotyczących specyficznych problemów związanych z rozpoznawaniem i interpretacją wyrażeń temporalnych. Badania te dotyczą rozpoznawania wyrażeń z wykorzystaniem informacji składniowej zdań, interpretacji wyrażeń zbudowanych w oparciu o nazwy dni tygodnia, oraz wyznaczaniem czasu odniesienia do interpretacji wyrażeń zależnych od kontekstu użycia.

Końcowa część rozprawy dotyczy technicznych aspektów konstrukcji systemu oznaczającego wyrażenia temporalne; w szczególności przedstawiony jest autorski system **DANTE**, który odznacza się znakomitą jakością. System ten jest dostępny publicznie w celu wsparcia dalszych badań wykorzystujących wydobywanie informacji temporalnej. Innym zasobem opracowanym w ramach przygotowywania niniejszej rozprawy jest korpus **WikiWars**, który również został udostępniony publicznie. Korpus ten zawiera pobrane z Wikipedii teksty narracyjne opisujące przebieg konfliktów zbrojnych, a występujące wyrażenia temporalne są oznaczone przy użyciu standardu TIMEX2. Dziedzina i gatunek tych dokumentów sprawiają, iż jest to jedyny w swoim rodzaju korpus dostępny w obszarze badań nad wyrażeniami temporalnymi.

Statement of Candidate

I certify that the work in this thesis entitled "Broad-Coverage Rule-Based Processing of Temporal Expressions" has not previously been submitted for a degree nor has it been submitted as part of requirements for a degree to any other university or institution other than Macquarie University (Australia) and Wrocław University of Technology (Poland).

I also certify that the thesis is an original piece of research and it has been written by me. Any help and assistance that I have received in my research work and the preparation of the thesis itself have been appropriately acknowledged.

In addition, I certify that all information sources and literature used are indicated in the thesis.

Signature: Paweł Mazur

Paweł Mazur – 41069846

12th December 2011

Acknowledgements

This thesis would not be in the state in which it is, and, realistically, would not have been possible at all, without the support of many people.

First of all, I have been very fortunate to have wonderful supervisors. Foremost, I am indebted to my supervisor at Macquarie University, Professor Robert Dale, for the inestimable amount of effort and time he has spent to provide me guidance in carrying out high-quality research and scientific writing. I am very thankful for all his comments and suggestions, and for the numerous discussions we have had. During the time I have spent working with Professor Dale I have learnt about many various aspects of being a researcher; this has undoubtedly contributed to fulfilling in excess the goal of making the PhD studies a training in research. Moreover, Professor Dale has been an excellent teacher not only with regard to the subject of my PhD, but also a mentor of how to be a great human being. I also appreciate the assistance of Professor Mark Dras, his support in administrative processes at various stages of the PhD program and for the numerous occasions he created to prove that doing research can be a socially rich and pleasant experience. Last, but not least, I would like to express my gratitude to Professor Zbigniew Huzar at Wrocław University of Technology for his support, feedback and words of encouragement.

I acknowledge the support of the Defence Science and Technology Organisation (DSTO), which funded a project in which I had the opportunity to participate; this introduced me to the topic of the processing of temporal expressions and helped me choose the area of research for my thesis. At this point, I should also not forget about Maciej Piasecki from Wrocław University of Technology who introduced me to natural language processing before I started my PhD program, and Leszek Maciaszek, who put me in touch with Macquarie University, offered his kind friendship and showed that every situation has its bright side.

I thank my colleagues from Macquarie University for their friendship and warm welcome *Down Under*: Elena Akhmatova, Steve Cassidy, Robert Dale, Mark Dras, Dominique Estival, Mary Gardiner, Andrew Lampert, Vanessa Long, Diego Mollá, Cécile Paris, Joanne Pizzato, Luiz Pizzato, Brett Powley, Jean-Philippe Prost, Rolf Schwitter, Marc Tilbrook, Jette Viethen, Stephen Wan, Mohammed J. Yaghi, Menno van Zaanen, and Simon Zwarts. In the later stage of my work on the thesis I also had the great pleasure of getting to know new members the Centre for Language Technology: Ilya Anisimoff, Matthew Honnibal, Suzy Howlett, Mark Johnson, Teresa Lynn, Alexandre Rafalovitch, Rainer Wasinger, Jojo Wong, and a number of visitors. All these people contributed to the unique atmosphere that made me enjoy doing my research and preparing the thesis.

Last but not least I would like to thank my closest ones, that is my parents, my brother and my wife, who since the beginning of my PhD program believed in me and supported in many different ways. I am blessed to have parents who have always found time and energy to be with me when I needed them and who have never let me down. And I am obliged for the patience that Daria had when she waited long years while I travelled across the world and time-zones to pursue my research interests and career.

Publications

Work presented in this thesis has received positive feedback from the research community, which is reflected in positive review comments leading to acceptance of research papers at national and international conferences and articles in research journals.

The summary of the work involved in the preparation of the WikiWars corpus, presented in Chapter 3, can be found in the following paper:

- P. Mazur and R. Dale [2010] WikiWars: A New Corpus for Research on Temporal Expressions. In the *Proceedings of the EMNLP 2010, Conference on Empirical Methods in Natural Language Processing*, 9th–11th October 2010, MIT Stata Center, Massachusetts, USA.

The syntactic method for extent recognition based on functional dependencies, presented in Chapter 5, has been published in the following paper:

- P. Mazur and R. Dale [2011] Temporal Expression Recognition Using Dependency Trees. In the *Proceedings of the 5th Language & Technology Conference (LTC)*, pages 141–145, 25th–27th November 2011, Poznan, Poland.

The work described in Section 6.2 about the LTIMEX extensions to the existing annotation schemes to encompass the representation of local semantics of temporal expressions has been presented in the following papers:

- P. Mazur and R. Dale [2006] An Intermediate Representation for the Interpretation of Temporal Expressions. In the *Proceedings of the COLING/ACL 2006 Interactive Presentation Sessions*, 17th–21st July 2006, Sydney, Australia, pages 33–36.
- R. Dale and P. Mazur [2006] Local Semantics in the Interpretation of Temporal Expressions. In the *Proceedings of the Workshop on Annotating and Reasoning about Time and Events (ARTE2)*, pages 9–16, 23rd July 2006, Sydney, Australia.
- R. Dale and P. Mazur [2007] The Semantic Representation of Temporal Expressions in Text. In the *Proceedings of the 20th Australian Joint Conference on Artificial Intelligence*, 2nd–6th December 2007, Gold Coast, Queensland, Australia. Springer-Verlag Lecture Notes in Artificial Intelligence (LNAI) series, vol. 4830/2007, pages 435–444.
- P. Mazur and R. Dale [2011] LTIMEX: Representing the Local Semantics of Temporal Expressions, In the *Proceedings of the 1st International Workshop on Advances in Semantic Information Retrieval (ASIR), Federated Computer Science & Information Systems Conference (FedCSIS)*, pages 201–208, 18th–21st September 2011, Szczecin, Poland.

The algorithm for interpretation of weekday names (see Section 6.4) has been published in the following paper:

- P. Mazur and R. Dale [2008] What's the Date? High Accuracy Interpretation of Weekday Names, In the *Proceedings of the 22nd International Conference on Computational Linguistics (Coling)*, 16th–24th August 2008, Manchester, UK, pages 553–560.

Overall presentation of the DANTE system, described in Chapter 7, with its evaluation at the time of its development, can be found in the following papers:

- P. Mazur and R. Dale [2007] The DANTE Temporal Expression Tagger. In the *Proceedings of the 3rd Language & Technology Conference (LTC)*, 5th–7th October 2007, Poznan, Poland.

  An extended and improved version of the above paper was later published as a book chapter:

  P. Mazur and R. Dale [2008] The DANTE Temporal Expression Tagger. in Zygmunt Vetulani and Hans Uszkoreit (eds.) *Human Language Technology. Challenges of the Information Society*, pages 245–257 Lecture Notes in Computer Science, Vol. 5603, Springer.

- P. Mazur and R. Dale [2007] A Rule-Based Approach to Temporal Expression Tagging. In the *Proceedings of the International Multiconference on Computer Science and Information Technology (IMCSIT), 2nd International Symposium: Advances in Artificial Intelligence and Applications (AAIA'07)*, 15th–17th October 2007, Wisla, Poland.

  The above paper was later republished as a journal article:

  P. Mazur and R. Dale [2008] A Rule-Based Approach to Temporal Expression Tagging. In *Systems Science*, 34:4, pages 19–26.

The DANTE system has also been shown to the community at demonstration sessions at various conferences: COLING/ACL 2006, IMCSIT/AAIA 2007 and LTC 2007. We have also participated in the ACE 2007 TERN evaluation task, which resulted in the following ACE workshop paper:

- P. Mazur and R. Dale and M Milosavljevic [2007] The DANTE Temporal Expression Tagger. In the *Proceedings of the ACE 2007 Workshop*, 28th–29th March 2007, Washington D.C., USA.

Some of the above papers have been cited by the following authors: Ahn et al. (2007), Niemi and Koskenniemi (2008b), Parent et al. (2008), Saquete et al. (2009), Hripcsak et al. (2009), Elkhlifi and Faiz (2010), Li et al. (2010), Derczynski and Gaizauskas (2010), Scott (2010), Teissèdre et al. (2010), Northwood (2010), Waring (2010), Battistelli et al. (2011), Janarthanam et al. (2011), Siabato and Manso-Callejo (2011), Derczynski and Gaizauskas (2011), Alonso et al. (2011), Saquete and Pustejovsky (2011), Strötgen and Gertz (2011a), Strötgen and Gertz (2011b), Kumar et al. (2011), Wonsever et al. (2011).

Acronyms and Notational Conventions

Acronyms

ACE	Automatic Content Extraction
ADV	adverb
ADVP	adverbial phrase
AVM	attribute–value matrix
DCD	document creation date
FST	finite-state transducer
IE	Information Extraction
ISO	International Organization for Standardization
JAPE	Java Annotation Patterns Engine
LDC	Linguistic Data Consortium
LHS	left-hand side
ML	machine learning
MUC	Message Understanding Conference
NER	named entity recognition
NIST	National Institute of Standards and Technology
NLP	Natural Language Processing
NP	noun phrase
POS	part-of-speech
PP	prepositional phrase
QA	question answering
RHS	right-hand side
TE	temporal expression
TERN	Time Expression Recognition and Normalization
TIDES	Translingual Information Detection, Extraction, and Summarization
UTC	the Coordinated Universal Time
VP	verb phrase

Notational Conventions

When used for the first time, **terms** of special significance in our work are presented in bold face. A ‘mention’ of a term is presented in quotes.

Linguistic examples and *titles of documents and specifications* occurring in the body of the text are in italics; the context should suffice to distinguish between these two categories.

Example text annotations, code fragments and names of XML tags and their attributes are printed using a `typewriter typeface`.

Abstract values are formatted with a Sans Serif font style.

Names and alphanumeric identifiers of documents contained in a corpus are formatted with SMALL CAPITALS.

‘Quotations in the body of the text’ are provided within single quotes.

Citation Conventions

It is not uncommon that a publication, especially one that was originally published decades ago, has been recently reprinted. In such cases, both the original source and the reprint are listed in the bibliography, but the citation in the text of the thesis refers to the earliest publication. For example, the original article *Verbs and times* was published by Vendler in 1957, then reprinted in 1967 and 2005. Although the only copy we have access to was from the collection from 2005, we cite the original work Vendler (1957), but the reader can find all three references in the bibliography.

When citing a published piece of work longer than a typical conference paper of a few pages in length we have tried to be quite specific about where the citation comes from by indicating the chapter, section or page number. In such cases the citation contains additional information after the publication year; for example, *Akmajian et al. (1990, Chapter 6)* or *Akmajian et al. (1990, pp. 255–260).*

Chapter 1

Introduction

The extraction of temporal information is important for many applications of natural language processing (NLP), such as information extraction, question answering, and text summarization. The full temporal analysis of a text is a very complex task, requiring a proper treatment of a wide range of linguistic and other phenomena including tense, aspect and causality. This thesis focuses on a key constituent task: the processing of linguistic expressions that refer to temporal entities such as points in time (e.g. *5 o'clock next Wednesday afternoon*) and temporal intervals (e.g. *the first two weeks of next year*). The reliable identification and interpretation of these signals of temporal information is an important, and surprisingly difficult, first step in carrying out the in-depth temporal analysis of text. We analyse in detail the forms that such expressions can take, and we investigate the practical aspects of recognizing and interpreting these expressions. From an engineering perspective, our ultimate goal is to construct a software system that can identify in text documents those linguistic expressions that describe points in times and intervals with both high accuracy and performance within reasonable time bounds, and can express their meaning in some predefined formal language or semantic representation. Such an implementation must, however, be supported by and based on a clear theoretical understanding of the underlying problems and sound algorithms for solving them; these aims provide the intermediate goals of our research.

1.1 The Problem: Processing Temporal Expressions in Texts

This thesis presents research on the topic of the processing of temporal expressions within texts, which involves both identifying time-denoting expressions and capturing their meaning. In the literature this task is usually referred to as **temporal expression recognition and normalisation**, or TERN for short. It is an interesting and challenging task because, while some temporal references appear in well-defined formats (e.g., dates like *17-07-1984*), others are expressed using a wide range of natural language constructions whose interpretations are context-dependent (e.g. *17 July*, *next Monday* or *two months later*); appropriate interpretation in these cases therefore requires an analysis of the surrounding text.

The TERN task is an important subtask in many natural language processing applications. To take question answering (see, for example, Maybury (2004)) as a case in point: consider Example (1.1) below, which presents an imaginary but plausible description of a journey from Sydney to Europe.

(1.1) We left Australia on a flight from Sydney on **Friday 14th July 2006**. We had booked it already on **15th December** to get a cheap fare. Our flight departed at **8:20pm**, and after **11 hours** in the air we arrived in Singapore, where we had arranged a stopover. Although we were tired, we spent **all of Saturday** walking around and taking photographs, trying to squeeze in as much as possible; our flight onwards to Frankfurt was due to depart **the following day**.

This text contains answers to a number of time-related questions that one might ask. A question might concern information stated directly in the text, as in Example (1.2a); or it might require some form of reasoning to obtain the answer, as in Examples (1.2b) and (1.2c). In all these cases, it is first necessary to recognise and properly interpret the temporal expressions in the source text.

(1.2)
a. *On what date did we leave Australia?* Answer: `2006-07-14`
b. *On what date did we book the flight?* Answer: `2005-12-15`
c. *On what date did we leave Singapore?* Answer: `2006-07-16`

Other natural language processing applications that can benefit from the processing of temporal expressions are document summarisation (see (Mani and Maybury, 1999) for a representative collection of papers from this area), information extraction (an extensive introduction can be found, for example, in (Moens, 2006)), textual entailment (see, for example, (Dagan et al., 2006)) and the temporal indexing of documents in search engines (see (Alonso, 2008) for an in-depth analysis of the problem). The correct processing of temporal expressions is also necessary for carrying out other time-related tasks, such as temporal reasoning (a broad account of the area and the applications to artificial intelligence can be found in (Fisher et al., 2005)), temporal information visualisation (a collection of a set of innovative solutions in this area have been collected by Shneiderman and Bederson (2003)) and the sequencing of events mentioned in documents (for example, an application to processing reports about car accidents has been studied by Berglund (2004)).

The extraction of temporal information can be carried out at different levels of sophistication and concern different elements of language analysis. In the broadest approaches, such as those taken by Smith (1978), Hirschman (1980) and Hinrichs (1986), an analysis of tense and aspect is required in order to analyse the absolute (i.e. on the timeline) and relative temporal locations of eventualities (i.e. events, processes, activities and states). The complexity of the analyses required means that much of this work, such as that described in Smith (1978) and Hinrichs (1986), is principally theoretical in nature, although Hirschman (1980) describes an implemented system capable of processing narrative medical records. Computational work undertaken in the information extraction community tends to be narrower in scope; analysis is limited to specific types of linguistic constructions (typically noun phrases, prepositional phrases and adverbials) which refer to points of time and intervals of time and their durations. This narrower task can be seen as a first step in the overall process of temporal information extraction and analysis; however, it is still far from being trivial. Temporal expressions can have numerous forms which make recognition difficult, and the context-dependent cases often require a sophisticated analysis of the surrounding text to derive their semantics.

Consider once again Example (1.1). It contains six temporal expressions, which we distinguished from the rest of the text using bold face. The first expression (*Friday 14th July 2006*) is context-independent, which means that the content of the expression itself is sufficient to determine its meaning. The other five expressions are context-dependent. The first, *15th December*, requires the year component to be filled in. To do that correctly it is necessary to observe that the expression is related to a flight booking which occurred before the flight; consequently the year must be 2005, rather than 2006. The time expression *8:20pm* must be assigned to the previously found date `2006-07-14`,[1] which we refer to here as the **reference time**, a term we will use to refer to a privileged absolute point in time that can be used to compute the value of other points in time. The information contained in the duration expression *11 hours* can be used in further reasoning if it is attached to its beginning anchor `2006-07-14T20:20`. The last two expressions ultimately refer to dates, but in two substantially different ways: *all of Saturday* is underspecified and based on a weekday name, while *the following day* is an offset from some other contextually-determined temporal value. The text does not say directly which Saturday was meant by its author, and interpretation of this expression requires some sort of reasoning and calendar knowledge. A key factor in each case of a context-dependent expression is the correct choice of its reference time.

Amongst context-dependent temporal expressions we identify **underspecified** expressions; they differ from context-independent expressions by missing some date or time component. We have three such expressions in Example (1.1): *15th December* (missing a year), *8:20pm* (missing a year, month and day), and *all of Saturday* (missing a year and a week within the year). All these expressions are based on cyclical elements of a calendar, and apart from requiring reference time, they additionally require that the **direction of interpretation** from the reference time is determined. For example, *15th December* is located in the past of its reference date (`2006-07-14`), *8:20pm* is placed within the reference date (`2006-07-14`), and *all of Saturday* is in the future of the reference date (`2006-07-14`).

In the work presented in this thesis, our aim is to develop a model that can represent the different types of temporal expressions generally considered within-scope for the information extraction approach mentioned above. We look at expressions occurring in different genres of texts, such as news, conversation transcripts, historical texts, web blogs, discussion forums, legal documents and emails. A key goal is a well-grounded implementation of a software component with high accuracy, and robust enough to be

[1]We are using here the ISO 8601 format to express absolute values of dates and times.

useful for processing large collections of real-world documents.

Such a component should be able to read in a document, such as the one presented in Figure 1.1, and output information, in either in-line or stand-off fashion, about temporal expressions found in the document and their semantic interpretations. Figure 1.2 presents a corresponding output document with in-line annotations using the TIMEX2 scheme (Ferro et al., 2005). Here, for example, interpretation of the interval-referring string *four years* results in determining that it refers to a period of four years (`val="P4Y"`), which ended in year 2003 (`anchor_dir="ENDING" anchor_val="2003"`). Of course, one could infer more precisely that the four year period in fact ended on 4th April 2003, or some earlier date very close to it.[2] However, the TIMEX2 scheme used here assumes annotations with semantic values at the granularity of the temporal units of the lexical items used in the expression; here, a year. This example already demonstrates that the TERN task can be specified in many different ways; it may be more or less precise, it may or may not require contextual reasoning, and so on. In fact, the particular application context may determine what is considered to be a temporal expression, how deep the analysis of its meaning should be, and how the semantics should be represented. As a consequence, the comparison of performance of different temporal information extraction systems must take into account how the TERN task was defined for their construction and evaluation.

In the literature concerning the processing of temporal expressions from an information extraction perspective, e.g. as in the Automatic Content Extraction (ACE) program[3] or other works based on the TIMEX2 specifications, the TERN task is often presented as one involving two subtasks: the **recognition** of temporal expressions and their **normalisation**. The recognition task is concerned with identifying occurrences of temporal expressions in a text and determining their **extents**, i.e. finding which text tokens constitute the expressions. The normalisation task requires that the temporal information contained in each expression be extracted and encoded in a uniform representation. Both of these tasks need to be satisfactorily addressed by any solution that attempts practical temporal expression processing.

1.2 An Overview of the State-Of-The-Art

In recent years, there has been increased interest in the processing of temporal expressions, from both theoretical and practical perspectives. We can distinguish three major areas of activity: the representation of temporal information, the development of annotation schemes and associated corpora, and temporal expression processing. We briefly outline the current state-of-the-art here, and in Chapter 2 we further elaborate on each of these areas.

With respect to the representation of temporal information, we can distinguish two dimensions: the taxonomisation of temporal expressions, and the development of frameworks for semantic representation.

Taxonomisation is concerned with characterising the different types of temporal expressions that are found in texts. In the information extraction community, it is generally accepted that there are two kinds of temporal entities: points and periods.

[2]The document, published on 2003-04-04, reports in the present perfect tense about an event (acceptance of a loan) which is the end point of a process (some negotiations) that lasted four years. Associating the publication date with the acceptance event determines the end of the four-year period.

[3]See `http://www.itl.nist.gov/iad/mig//tests/ace/2004/doc`.

MOSCOW, 2003-04-04

Russia has accepted a US$150 million World Bank loan to combat the spread of AIDS and tuberculosis, ending a negotiating process that lasted four years, World Bank officials said Friday.

The World Bank first offered the loan in 1999, but disagreements over treatment kept the project on hold. Russia objected to World Bank rules that required monitoring of patients receiving medication, the World Bank said.

But after drawn-out talks, President Vladimir Putin signed off on the loan, which will take effect this year, the World Bank said.

Figure 1.1: A fragment of an example input document.

MOSCOW, <TIMEX2 val="2003-04-04">2003-04-04</TIMEX2>

Russia has accepted a US$150 million World Bank loan to combat the spread of AIDS and tuberculosis, ending a negotiating process that lasted <TIMEX2 val="P4Y" anchor_dir="ENDING" anchor_val="2003">four years</TIMEX2>, World Bank officials said <TIMEX2 val="2003-04-04">Friday</TIMEX2>.

The World Bank first offered the loan in <TIMEX2 val="1999">1999</TIMEX2>, but disagreements over treatment kept the project on hold. Russia objected to World Bank rules that required monitoring of patients receiving medication, the World Bank said.

But after drawn-out talks, President Vladimir Putin signed off on the loan, which will take effect <TIMEX2 val="2003">this year</TIMEX2>, the World Bank said.

Figure 1.2: A fragment of an output document with inline TIMEX2 annotations.

Temporal expressions may refer in a variety of ways to a single entity or to a set of entities of a given type. There have been several taxonomies developed by different authors but none of them has been adopted as a standard and the terminology used in this area varies considerably; there is a pressing need for consolidation.

Many frameworks have been developed for semantic representation, but three of them are particularly worth mentioning in the context of our work as they were developed specifically to capture the meaning of temporal expressions: (i) the OWL-Time ontology constructed by Hobbs and Pan (2004) and further extended by Pan (2007), (ii) a compositional semantics developed by Schilder (2004), and (iii) Han and Lavie's (2003; 2004) Time Calculus for Natural Language (TCNL). These are based on various formalisms: OWL-Time is defined in first-order logic and can be implemented in logic-based markup languages, such as OWL; Schilder's compositional semantics is based on the lambda calculus; and TCNL is built on top of a constraint-based model for human calendars. All these representation frameworks have reasonably wide coverage and have been successfully used in practice; OWL-Time has been applied to describe the temporal content of web pages and the temporal properties of web services and the other two underpin existing temporal information extraction systems.

An annotation scheme specifies how expressions should be annotated in text, and as part of this it may indicate what semantic representation should be included in the annotations. In theory any representation can be used, but when designing an annotation scheme there is a trade-off between readability and expressiveness. Two major standards developed for information extraction are currently being used to annotate temporal expressions: TIMEX2 (Ferro et al., 2005) and TimeML (Pustejovsky et al., 2005). In both cases a temporal expression is annotated in-line and the semantics is represented by a set of attributes making use of the ISO 8601 date and time representation format; a few example TIMEX2 annotations have already been presented in Figure 1.2. TIMEX2 is based on the TIMEX annotation style used at the Message Understanding Conferences (MUC) evaluations, with the addition of semantic representation to the annotations. The TimeML standard, originally built upon experiences both from TIMEX2 and the work presented by Setzer (2001), introduced annotations for events and relations. Both schemes cover a broad range of expressions, especially those occurring in news; however, they only provide a representation for the meaning once it has been interpreted in context. We will argue that a context-independent representation of meaning is also useful.

Annotated corpora are valuable resources for researching NLP-related problems, and for system development and evaluation. Given their importance, the Linguistic Data Consortium[4] (LDC) distributes a number of corpora annotated using TIMEX, TIMEX2 or TimeML. These include the datasets used at the MUC evaluations and the ACE TERN tasks, and also the TimeBank corpus (Pustejovsky et al., 2003; Boguraev et al., 2007). Apart from the corpora available via the LDC, there is also the TIDES corpus available from MITRE's website.[5] These corpora contain documents from a number of domains, mostly news, but also UseNet discussions, web blogs and transcripts of conversations on various topics; notably, none of them contains documents presenting protracted narratives which, as we will show, impose more sophisticated processing than the other types of documents do.

Ultimately, the purpose of much of this work in representation and annotation, and the subsequent development of corpora, is to support the development of software capable of carrying out automatic annotation of temporal expressions in texts. These systems may only identify the occurrence of temporal expressions or they may also analyse their semantics; in either case such a tool is often referred to as a **temporal expression tagger**. Most existing taggers make use of hand-coded rule-based grammars, which usually perform both the recognition and interpretation tasks, but some machine-learning (ML) approaches have also been presented in the literature (for example, by Jang et al. (2004), Hacioglu et al. (2005), and Ahn et al. (2007)). Although the first experiments using machine learning for automatic temporal expression tagging were reported just over a decade ago by Mani and Wilson (2000b), the development of ML-based implementations on a wider scale would not be possible if not for the appearance of the ACE corpora and the TimeBank corpus. Taggers based on machine-learning algorithms have mainly been applied to the recognition task only (see, for example, the description of the ATEL system developed by Hacioglu et al. (2005)), but there have also been attempts made to use machine learning in the interpretation step (see the work of Ahn et al. (2007)). As we mentioned earlier, the evaluation of a temporal expression tagger depends in large part on how the notion of a temporal expression is defined in the given application, and in consequence it is not always possible to directly compare two taggers just by looking at their raw performance indicators as measured using one of the common metrics. While the existing taggers perform very

[4]See `http://www.ldc.upenn.edu`.

[5]See `http://timex2.mitre.org`.

well in many cases, and especially in the news domain, there is room for improvement in many more difficult cases.

1.3 What is Still Missing?

The ultimate goal of the machinery developed in the temporal information extraction community is to process text documents written in a natural language in order to extract the temporal information they contain, and to represent this information in some standard, well-structured format for the purpose of exchange between machines and further processing. Although in recent years we have observed significant advances towards this goal, there are still many areas in which further work is needed and improvements required.

The concept of time has long been a subject of study in disciplines such as philosophy, mathematics, physics and logic. In the past few decades it has also received a lot of attention from linguists. While it would be hard to claim that all the problems have been solved, we can say that there is now a great deal of analysis, and many proposed theoretical models of time. However, in the era of computer-based information processing, we now face the problem of how to implement these theories to create robust, efficient and highly accurate software systems that can process large amounts of text. On the one hand we have very sophisticated formal approaches, which attempt to cover the full diversity of temporal phenomena found in the real-world; but these generally remain purely theoretical descriptions. On the other hand we have a range of software implementations that process temporal information, but very often these perform well only in limited domains, using shallow text processing and generally covering only the more common and frequent kinds of temporal expressions. It is our view that further advances in improving results are to be achieved by making software systems more sophisticated, so that they cover not only the most common structures, but also those which are less common and often much more challenging.

Problems and limitations arise from a number of factors. The practical requirements of evaluation and data exchange, for example, often lead to the use of annotation schemes which involve simplifications that are less than ideal. Many of the existing implementations are not always founded on more theoretically well-grounded taxonomisations and formal representations. Yet another issue is that many systems documented in the publicly available literature were prepared primarily for the purpose of participation in evaluation programs such as ACE. In such cases, the pragmatic issues that must be addressed and pressures that must be faced can lead to systems being tailored to specific domains and text genres, or being focused on the most frequent cases that appear in the training data, with a risk of sidestepping deeper analyses of the underlying problems to be solved.

Addressing the shortcomings in the current state-of-the-art requires us to advance our knowledge in the following areas:

1. The taxonomization of temporal expressions, with the purpose of classifying and describing the full range of temporal expressions that are found in texts, and the development of a representation of their semantics which is appropriate for implementation in software.

2. The development of annotation schemes, with the aim of providing annotations

capable of expressing the required semantics, and the construction of annotated corpora, to provide development and evaluation datasets.

3. The design and implementation of algorithms for temporal expression tagging which are robust, efficient, perform well, and provide wide coverage of the temporal expressions found in text, including their semantic annotation.

The work presented in this thesis aims to address all of these areas.

1.4 The Aims of this Work

The main motivation for the work presented in this thesis is a desire to improve the performance of technology for identifying and interpreting temporal expressions. We aim to achieve this goal by carrying out an analysis of the problems of recognition and interpretation of temporal expressions, identifying the underlying difficulties, and designing relevant algorithms.

As a first step, based on previous work in this area and our empirical experience, we define a comprehensive taxonomy of temporal expressions. An additional outcome of the empirical studies is the creation of a new annotated corpus, whose originality and usefulness lies in its domain being substantially different from the domains of existing annotated corpora, and the presence of phenomena under-represented in those corpora.

The initial task in the successful processing of a temporal expression is the correct recognition of its occurrence in a text. We investigate this problem and explore new techniques based on using syntactic information, which until now has been neglected in the existing literature.

We also propose a representation for what we call the **local semantics** of temporal expressions. The aim is to make this representation compatible with the TIMEX2 and TimeML standards, thus introducing a useful intermediate level of semantic representation in temporal expression annotation.

Different types of temporal expressions require different approaches to interpretation; based on our taxonomy of these expressions, we aim to provide a range of algorithms which can correctly interpret expressions of the different types. A common and significant problem in the interpretation of context-dependent expressions is the correct selection of the reference time, a task that we will call **temporal focus tracking**. Another problem is the choice of the **direction of interpretation** (i.e. determining whether the referred-to temporal entity is located in the past or in the future from the reference time, or is a differently-grained view on the reference time) for underspecified temporal expressions, such as bare weekday names. Our objective is to experiment with various approaches to these problems.

Finally, the combination of the outcomes of our analyses and experiments results in the creation of a computer system capable of finding temporal expressions in texts and providing a formal interpretation of these expressions. This system is expected to stand out with high accuracy and to be robust enough to be useful for processing real-world documents.

1.5 The Contributions of the Thesis

Overall, this thesis makes a number of contributions to the processing of temporal expressions, and extends the body of knowledge in this area both at the theoretical and engineering levels.

First, we provide a comprehensive review of the current state-of-the-art in the area of processing temporal expressions. This part of the thesis may serve as a useful reference source for someone new to this research field.

Next, we present a detailed taxonomy of temporal expressions. This results from analyses of the expressions found in texts, and unifies a number of taxonomies that are found in the literature. We hope that this consolidation will make it easier to discuss the open research questions and support comparison of results.

In the area of extent recognition, we demonstrate that syntactic information can be successfully used in the recognition of temporal expressions, and in particular in the case of expressions of complex syntactic structure.

In our work we clearly separate the representation of local and global semantics of temporal expressions. Since annotations in both TIMEX2 or TimeML support only the latter, we designed a representation for local semantics, which we called LTIMEX, that is compatible with these annotation schemes. This introduces the possibility of enriching existing annotated corpora with an additional level of annotation which would be useful for development and more detailed evaluation of temporal expression taggers. LTIMEX may also serve as an interface between different software modules generating the local and global semantics.

The thesis also presents the first comparative evaluations of approaches to addressing temporal focus tracking and bare weekday name interpretation. In both cases we evaluate a number of possible heuristics proposed in the literature but not evaluated properly before, and develop variations of these approaches resulting in the best accuracy measured on the evaluation datasets we use.

From an engineering perspective, we present a new tagging system, named DANTE, which implements our approach to processing temporal expressions. The system provides broad coverage and state-of-the-art accuracy and performance. We make the system publicly available as a plugin to the GATE platform, which should be of help to researchers working in other areas of natural language processing, such as question answering or text summarisation, where the extraction of temporal expressions is a prerequisite.

We also make publicly available our WikiWars corpus, containing over 2680 TIMEX2 annotations in documents describing the temporal progression of military conflicts. We hope that this corpus will foster further research in this area, particularly on temporal focus tracking and the interpretation of event-based temporal expressions.

1.6 The Structure of the Thesis

The remainder of this thesis is structured as follows.

In Chapter 2 we review the existing literature on the processing of temporal expressions. We begin by discussing temporal ontology, introducing a number of key concepts which we use throughout the thesis; we then review work on the taxonomisation of temporal expressions, the semantic representation of temporal information, annotation schemes and existing annotated corpora, and existing approaches to the automatic temporal tagging of text.

In Chapter 3 we introduce our annotated WikiWars corpus, which we developed to explore phenomena that are under-represented in the currently available annotated corpora.

In Chapter 4 we present the taxonomisation of temporal expressions that underlies our work; this unifies and systematises a number of taxonomisations found in the literature, and provides a foundation for the design of recognition and interpretation algorithms.

Next, in Chapter 5, we go on to explore whether syntactic parsers can be successfully employed in the recognition of temporal expressions. We experiment with dependency-based and constituency-based analyses of sentence structure provided by a range of state-of-the-art parsers.

In Chapter 6 we turn to the interpretation task. In our model, the semantic interpretation of temporal expressions is separated into two distinct stages, concerned with what we refer to as local and global semantics. The former is a context-free representation of meaning based on the recognized string alone; the latter adds contextual information to provide a fully-fleshed out representation of meaning. We first present our flat representation of local semantics, which is compatible with existing annotation schemes such as TIMEX2 or TimeML. Then, we explore a number of specific issues in global semantic interpretation, including the handling of calendar cycles in the interpretation of bare weekday names and temporal focus tracking.

Based on the foregoing discussions, the goal of Chapter 7 is to present our temporal expression tagging system, DANTE. We discuss its architecture, implementation, and performance; the system is evaluated on a number of available annotated corpora and compared with other systems presented in the literature.

We conclude in Chapter 8 by summarising the key achievements and results of the thesis, and outlining plans for future work.

Appendices provide additional information concerning development and evaluation issues.

Chapter 2

A Review of the Literature

In this chapter we present a review of the literature concerned with the processing of temporal expressions. First of all, in Section 2.1, we outline as background a basic taxonomy of temporal concepts and discuss what the term ‘temporal expression’ refers to. Then in Section 2.2 we review the types of temporal expressions identified in the literature, indicating the differences in terminology different authors use.

When discussing the representation of temporal expressions, we distinguish two levels: the semantic representation of an expression independent of any contextual information, and the fully worked-out representation of the temporal value to which the expression refers. In Section 2.3 we look at how existing temporal expression taggers deal with these two levels.

A very useful resource for conducting research in the area of natural language processing is an annotated corpus. Such a collection of text documents can serve us in two ways: as data to be used in the development of algorithms and techniques, and as data to be used in evaluation of those algorithms. So, in Section 2.4, we review schemes for the annotation of temporal expressions; then, in Section 2.5, we explore a variety of existing corpora that are marked up using these schemes.

Finally, in Section 2.6 we review the existing temporal expression taggers described in the literature. As in many other applications of natural language processing, we can observe here two kinds of approaches taken. One is knowledge-based, which requires an engineer to manually write a recognition grammar and interpretation rules. The other approach is based on machine-learning algorithms which, when trained on annotated corpora, produce a model which can be then used to process unseen raw text. Both approaches have their own strengths and weaknesses, which we summarise.

2.1 Defining Temporal Expressions

2.1.1 Temporal Ontology

To represent time-related phenomena we need an ontology of temporal concepts; this allows us to model situations happening in the world, and to provide mechanisms for reasoning about the temporal relationships that hold between these situations. Based on the literature, we identify three ontological categories that need to be considered: (a) temporal entities, (b) eventualities, and (c) temporal relations.

2.1.1.1 Temporal Entities

When thinking about how to represent time we can identify two types of temporal entities: **instants** and **intervals**. These are the primitive individuals—or atoms of time, as Galton (1995) calls them—which can be used as the foundations of complex systems for representing and reasoning about time. We note that a variety of terminology is used in the literature when referring to these types of temporal entities. An instant is also referred to as a **point in time**, a **time point**, a **point** or a **moment**. Van Benthem (1983, p. 240) prefers to use the term **period** instead of interval, since for him the term 'interval' refers to 'what lies between boundaries'; this is what others, for example Allen (1995), call **duration**. We will consider the terms 'period' and 'interval' interchangeable, and treat duration as a feature of a period.

Instants differ from intervals by having no duration, which in consequence means that two instants cannot overlap or be contained in one another. In other words, two instant situations happening at the same time in fact happen in the same instant. Intervals have duration, and therefore we can say they have some internal structure (i.e. we can identify subintervals) and distinguish many different relations between intervals.

There has been an ongoing dispute as to which entity type is more appropriate as the primary notion for a theory of time. Van Benthem (1983) claims that instant-based representations are counter-intuitive for modeling time, which, in his view, is a continuum. He claims that we cannot experience a point situation in every-day life (a point in time is an abstract notion and it has no duration); further, human languages do not provide any expressions that refer to points. Therefore, in his view, interval-based theories are more natural and better-suited to describing the world. This is not an idiosyncratic view; it has also been expressed by a significant number of researchers in the field of linguistic sciences, perhaps best represented by the well known work of Dowty (1979).

In discussions as to whether an interval-based approach is more suitable than a point-based theory, van Benthem (1983, pp. 5–6, pp. 54–55) does not exclude the possibility of having an approach based on both types of temporal entities. In this vein, Vilain (1982) developed a computer system for reasoning about time which was primarily based on the logic of intervals, but was extended with new primitive relations and new composition rules over these primitives so that it also covered the logic of point objects. Also the more recent work carried out for the purposes of representing information on the semantic web and processing natural language, and which resulted in the OWL-Time ontology (Hobbs and Pan, 2006), uses notions of both time instants and intervals.

2.1.1.2 Eventualities

The term **eventuality**, being a general term for any type of situation (something that happens, occurs, lasts or takes place), was popularised by Bach (1986). However, his typology of eventualities was not the first of this kind. The origin of such categorization can be seen in the work of Vendler (1957), who, refining some partial findings in this area, presented a four-fold distinction of verb types (called by Vendler **time schemata**): **activity**, **accomplishment**, **achievement** and **state**. Based on the grammatical criterion of whether a verb admits a progressive form, the first two types belong to one genus, and the other two types form another genus. A brief characterisation of the time schemata along with examples is as follows:

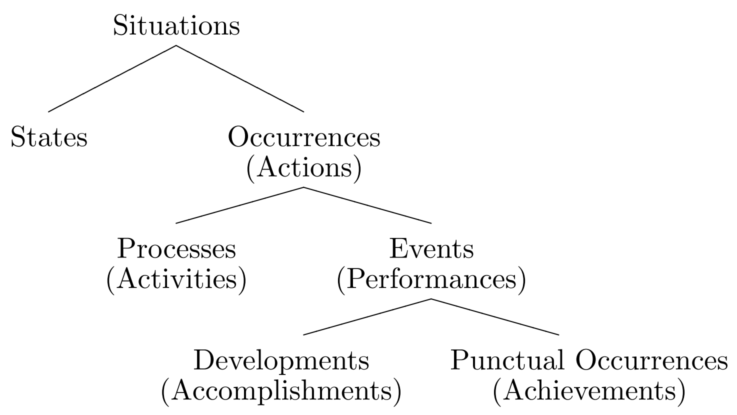
- An **activity** is homogenous (any part of the process is of the same nature as the whole) and has no climax or result; for example: running, walking, swimming, pushing or pulling something. Activities are also referred to as **processes**.
- An **accomplishment**, in contrast to an activity, is not homogenous and has a terminus. Examples of accomplishments are: running a mile, writing a letter, painting a picture, building a house, growing up, and recovering from illness.
- An **achievement**, although it takes some time, occurs at a single moment and usually captures either the inception or the climax of an act. Examples include: recognising, spotting something, losing or finding, winning the race, starting, stopping, resuming, being born, and dying.
- A **state** endures over a period of time (possibly very short or long), but it is not an action at all, and does not involve any dynamics, yet it may be predicated with truth or falsity: consider, for example, having, desiring, wanting, loving, hating, ruling, dominating.

A given verb in its different uses can refer to more than one time schema. For example, *think* can be used to refer to a process (*He is thinking about Jones*) or to a state (*He thinks that Jones is a rascal*).

Vendler's division was later augmented by Kenny (1963, Ch. 8) who treated accomplishments and achievements as one type called **performance**, since they both involve some kind of a product or outcome. Later, Mourelatos (1978) argued that Vendler and Kenny, both coming from the philosophical community, did not realise that the distinctions they made involved verb aspect, a phenomenon that had already been studied by linguists. Mourelatos argued that the analysis of just the semantics of individual verbs is not sufficient for coming up with a verb typology, and showed a number of problems in Vendler's and Kenny's approaches. Using new criteria involving the notions of aspect and countability, he proposed the ontology of situations presented in Figure 2.1.

Bach's (1986) typology of eventualities extended Mourelatos' structure by distinguishing two subtypes of states (**dynamic** and **static**) and two subtypes of achievements (called by Bach 'momentaneous events'): **happenings** and **culminations**; see Figure 2.2. The significance of Bach's work lies in describing an event algebra: a means of expressing structures of events and processes which have subevents and subprocesses (for example, the process of building a house involves the subprocess of pounding a nail).

In the area of work related more directly to ours, we note that Pustejovsky (1988) used the term **event-type** instead of 'eventuality' or 'situation'. In the account of



Examples:

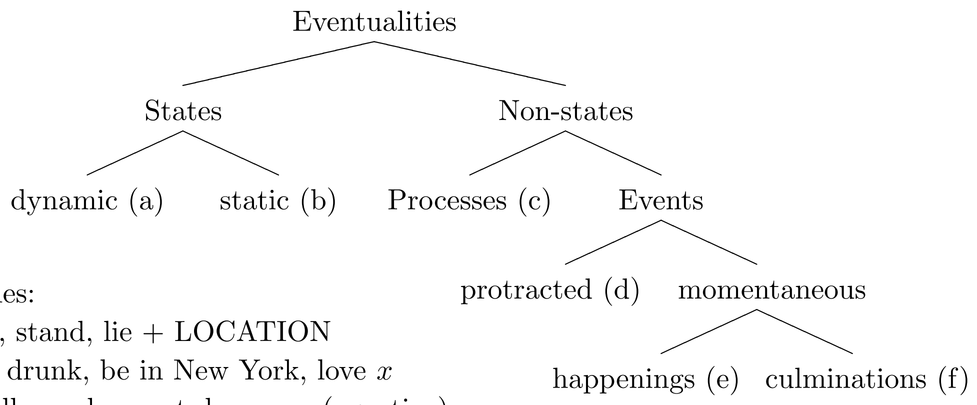
State:
The air smells of jasmine.

Process:
It's snowing.

Accomplishment:
The sun went down.

Achievement:
The pebble hit the water.

Figure 2.1: Ontology of Situations (Mourelatos, 1978).



Examples:
(a) sit, stand, lie + LOCATION
(b) be drunk, be in New York, love x
(c) walk, push a cart, be mean (agentive)
(d) build x, walk to Boston
(e) recognize, notice, flash once
(f) die, reach the top

Figure 2.2: Ontology of Eventualities (Bach, 1986).

Mani (1998), 'event' is a wider notion which also includes 'process', and 'achievement' is a subtype of 'accomplishment'.

Further distinctions in the ontology of situations have been made by, for example, Moens and Steedman (1988), Passonneau (1988), Masolo et al. (2003), and Palmer et al. (2007). Hajnicz (1996, Section 1.1) reviews the ontological concepts in the context of temporal reasoning in artificial intelligence.

2.1.1.3 Temporal Relations

Providing a formal representation of temporal relations is necessary if we want to obtain more complete models of the world. A knowledge base which contains only information about the occurrences of eventualities may be useful for some applications, but adding information about the relations that hold between them obviously extends the usefulness of the knowledge base.

A fundamental relation that can hold between two instants is the relation of precedence, marked as '$<$'. Probably the most cited work on relations between intervals is that due to Allen (1983). He distinguishes 13 temporal relations: identity (equals) and

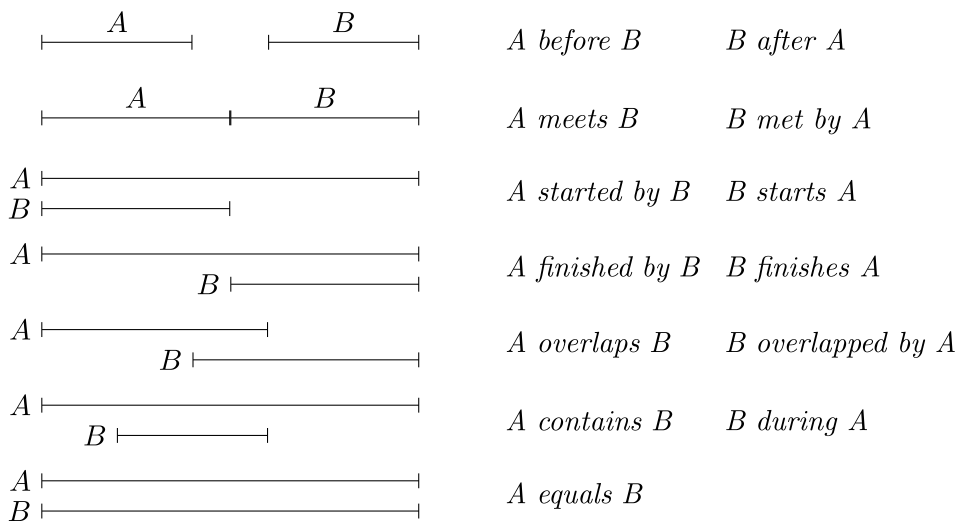


Figure 2.3: Allen's thirteen temporal relations between intervals.

six symmetrical relations, as follows:

- before/after: time interval A is before interval B and they do not overlap in any way;
- meets/met-by: time interval A is before interval B, but there is no interval between them (i.e. A ends where B starts);
- starts/started-by: time interval B shares the same beginning as A, but ends before A ends;
- finishes/finished-by: time interval B shares the same end as A, but begins after A begins;
- overlaps/overlapped-by: time interval A starts before B and they overlap; and
- during/contains: time interval B is fully contained within A.

These relations are graphically presented in Figure 2.3. Allen further provided an interval-based deduction technique for reasoning about relationships between intervals if new facts are added to the knowledge base. Examples of transitivity axioms used in this technique are:

$$\mathit{before}(A,B) \land \mathit{before}(B,C) \Rightarrow \mathit{before}(A,C)$$

$$\mathit{meets}(A,B) \land \mathit{during}(B,C) \Rightarrow \mathit{overlaps}(A,C) \lor \mathit{during}(A,C) \lor \mathit{meets}(A,C).$$

Temporal relations between eventualities can be analysed in terms of the relations between the temporal entities underlying these eventualities. For example, the temporal relations that hold between instances of riding a bicycle, falling and being bruised would be expressed as relations between three time periods: the riding meets the falling, and the falling is before the bruising; and using Allen's transitivity actions we could deduce that the riding has to be before the bruising.

2.1.2 What Constitutes a Temporal Expression?

In this section we discuss what might be meant by the term **temporal expression** and the differences there are in the use of the term in the literature. It turns out there is not, in the research community, one commonly agreed definition of what a temporal expression is. Some authors use the term in a very broad sense, while others use it in a quite restricted way. There are cases where, at first glance, two authors appear to talk about the same range of temporal expressions, but in practice there are subtle but significant differences in the definitions they are using.

2.1.2.1 The Broad Approach

We start with the broadest definition, which considers any fragment of a text expressing some temporal information to be a temporal expression. This includes analysis of tense, aspect, modals and auxiliary *have*, time and frequency adverbials, temporal noun phrases, and adverbial sentences. Providing a full temporal representation of the content of a document would often additionally require applying some mechanisms of reasoning to discover relations and facts not expressed directly in the text. Arguably, such reasoning is beyond the area of natural language processing per se, and is really a task for artificial intelligence.

Analysis of tense and aspect is crucial if we want to know in what time-frame a situation is placed. Both tense and aspect are reflected in verbs, and for this reason verbs could be considered to be temporal expressions. Work in this area is represented, for example, by the studies carried out by Reichenbach (1947), Smith (1978), Mourelatos (1978), Richards (1982), and Hwang and Schubert (1994).

A topic related to the analysis of tense and aspect is the study of modals which change the grammatical mood of a sentence; consider Examples (2.1c)–(2.1f). Identification of the differences is crucial for the correct processing of information about eventualities; it is important to recognize that none of these sentences states that an event happened: Examples (2.1c) and (2.1d) express a potential situation, and Examples (2.1e) and Examples (2.1f) use a conditional mood.

(2.1)
a. He *forgot* to reply to letters.
b. He *forgets* to reply to letters.
c. He *might forget* to reply to letters.
d. He *might have forgotten* to reply to letters.
e. He *would forget* to reply to letters.
f. He *would have forgotten* to reply to letters.

Further temporal information can be expressed by the use of adverbials (adverbs, adverbial phrases, and adverbial clauses) and noun phrases. Example expressions are listed in Table 2.1. Some words, such as *today* or *yesterday*, can be used either as adverbs (as in *I will do it tomorrow*) or as nouns (as in *Tomorrow will be a sunny day*). A distinguished group of temporal adverbs are adverbs of frequency; these can be considered to be positive (e.g. *often*, *always*), negative (e.g. *never*, *hardly ever*) or neutral, being somewhere between positive and negative (e.g. *sometimes*). Some frequency adverbs are based on time units (e.g. *hourly*); their interpretation is that something is done or is happening once in a given temporal unit, e.g. once an hour.

A temporal adverbial phrase very often takes the form of a prepositional or postpositional phrase. The semantics of prepositions has often been studied in conjunction with tense and aspect, but it has also been the object of particular studies in linguistics and logic; see, for example, Pratt and Francez (1997).

There are also sentences which have temporal adverbials consisting of multiple syntactic constituents, as shown in Examples (2.2) and (2.3).

Category	Examples
Adverbs	simultaneously, currently, lately, today, yesterday, tomorrow
Frequency Adverbs	a lot, always, ever, frequently, hardly ever, never, normally, occasionally, often, frequently, rarely, sometimes, usually, hourly, daily, weekly or monthly
Prepositional Phrase	on Monday, at 5 o'clock, for one hour, in January, over many years, during the weekend, after the meeting, before 8pm, between 11am and 1pm, by Monday morning, since 1978, until January 2006, from 1939 to 1945, within one hour, following the meeting
Postpositional Phrase	five months ago, five months hence, five months on, the whole night through
Other Adverbial Phrases	later than ever before, at least five years, all spring, on Tuesday at noon
Adverbial Clause	when she saw the snake, as soon as I have any news
Noun Phrase	coming weeks, a beautiful morning, cold winters

Table 2.1: Examples of expressions carrying temporal information.

(2.2) a. John visited his parents *twice* *in two years*.

b. John learnt Japanese *for half an hour* *every morning* *for a month*.

c. John washed cars *from morning till night* *from June till August*.

(2.3) a. John had arrived *on Tuesday* *at noon*.

b. *On Tuesday*, John had arrived *at noon*.

c. *Last week*, John had arrived *3 days ago*.

d. John had arrived *at noon* *on Tuesday*.

e. *John had arrived *3 days ago* *last week*.

In some cases one constituent of such an adverbial can be moved to the beginning of the sentence, which results in, following the terminology of Smith (1978), a **distributed adverbial**. For instance, in Example (2.3a) the temporal adverbial consists of two prepositional phrases: *on Tuesday* and *at noon*. The first can be moved to the beginning of the sentence, without changing its meaning, resulting in Example (2.3b). As observed by Smith (1978, p. 84), the sentence (2.3c), despite having the same surface structure as (2.3b), differs because it has two temporal adverbials, not one. Smith also provides a diagnostic for checking whether in such sentences there is one or two adverbials: if it is possible to construct a sentence with the two constituents next to each other by moving the initial adverbial to the end of the sentence, as in (2.3d), there is only one adverbial; otherwise, as in (2.3e), there are two.

Finally, a temporal adverbial that constitutes a temporal expression may be realised as an adverbial clause; this is a part of a main clause which is introduced with a conjunction, such as *when*, *after*, *as soon as*:

(2.4) a. Mary screamed *when she saw the snake*.

b. We had the chocolate pudding *after we ate the lunch*.

c. I will call you *as soon as I have any news*.

d. We won't stop repeating <u>until</u> *you say it correctly.*

The final form a temporal expression may take is a noun phrase. These may be used in a sentence either as a subject (see Example (2.5)) or an object (see Example (2.6)).

(2.5) a. *Coming weeks* will reveal who was right about the crises.

b. *Five years* is quite a long period to do nothing.

c. *A beautiful <u>morning</u>* is greeting us with good news.

d. *Cold <u>winters</u>* reduce the number of insects in summers.

e. *<u>Future</u> elections* will give us a new leader.

f. *<u>Former</u> presidents* criticized the plans of reforms.

(2.6) a. The expedition to Africa will last *about five months.*

b. He spent *six and a half years* in prison.

c. I will call you *next week.*

d. The clock was showing *5 o'clock* when the first bomb was dropped.

For a noun phrase to be considered a temporal expression, it must contain at least one time-related constituent: for example, the underlined nouns in Examples (2.5c) and (2.5d) or the underlined adjectives in Examples (2.5e) and (2.5f).

2.1.2.2 A Narrower Approach

In the area of information extraction (IE), the recognition of expressions whose heads are verbs is associated with **event recognition** rather than with **temporal expression recognition** (see, for example, (Setzer and Gaizauskas, 2000)). The task definition of event recognition requires the finding of an occurrence of an event mentioned in the text (usually, only events of specific types are the subject of interest in a particular application, e.g. company takeovers, joint ventures, bankruptcy), and extracting some predefined attributes of the event, such as the agent, object or a time-stamp (date and time). Because of this time-stamping of events, the extraction of temporal information often accompanies event extraction; however, they are still two distinct IE tasks. One exception to such a view is the work of Schilder and Habel (2001), who classified temporal expressions into two categories: time-denoting (e.g. *Friday*) and event-denoting (e.g. *opened*).

In the information extraction approach, temporal expressions are constituted by adverbs, adverbial phrases, and noun phrases conveying temporal information; however, these fall within a narrower range than the cases discussed in the previous section (e.g. some adverbs are not taken into account). Also, the analysis of tense and aspect is much more limited, and used primarily to detect whether a reference to time concerns the past or the future.

What we vaguely referred to above as 'the information extraction approach' is in practice almost exclusively bound to the specific annotation guidelines provided by TIMEX2 (Ferro et al., 2005) and TimeML (Pustejovsky et al., 2005), which we discuss in detail in Section 2.4. For the purposes of using these schemes, a temporal expression is defined as an expression which 'tells us *when* something happened, or *how long* something lasted, or *how often* something occurs' (Ferro et al., 2005, p. 6). This general statement is backed up by a list of lexical items which are considered

Category	Lexical Triggers	Non-Triggers
Noun	minute, afternoon, midnight, day, night, weekend, month, summer, season, quarter, year, decade, century, millennium, era, semester, future, past, time, period, point	instant, jiffy, episode, occasion, tenure, timetable, reign, light year, megawatt hour, lifetime, history
Adjective	recent, former, current, future, past, daily, monthly, biannual, semiannual, daytime, daylong, onetime, ago, preseason, short-term, long-term	early, ahead, next, subsequent, frequent, perpetual, later, contemporary, simultaneous, preceding, previous, existing, modern
Adverb	currently, lately, hourly, daily, monthly, ago (+ adverbial forms of adjective triggers)	earlier, immediately, instantly, forthwith, meanwhile, heretofore, previously, next, beforehand, following, later, soon, sooner, shortly, eventually, occasionally, once, still, again, timely, whenever
Time noun/adverb	now, today, yesterday, tomorrow	
Number	3, three, third, sixties	
Proper name	Monday, January, New Year's Eve, Washing's Birthday, Solstice	
Pronouns	that, then, it (only pronouns that co-refer with a markable expression)	
Time patterns	8:00, 12/02/2000, 1994, 1960s	

Table 2.2: Examples of triggers and non-triggers for temporal expressions in TIMEX2.

that cold day	the next day	late last night
earlier that year	next summer	recent decades
numerous Saturdays	more than a month	no less than 60 days
just a year ago	only one hour long	its own future
the countrys future	just a year ago	only one hour long
five years old	a few weeks later	hours earlier
five days after he came back	three decades ago	the second-best quarter ever
months of renewed hostility	a historic day for the European enterprise	nearly four decades of experience

Table 2.3: Examples of temporal expressions in TIMEX2.

triggers for temporal expressions, i.e. words (e.g. weekday names) that signal a possible occurrence of a temporal expression, and syntax-based rules determining what is included in the **extent** of the triggered expression. There is no predefined list of triggers; the guidelines only provide a few examples of triggers and non-triggers (which we list in Table 2.2) and the heuristic that a trigger must convey a temporal unit or concept.

An example rule concerning the extent of a temporal expression states that leading prepositions are not included. Therefore in Example (2.7) the preposition *before* is not considered to be a part of the temporal expression.

(2.7) I phoned her before *5 o'clock*.

Not all researchers follow this approach; Schilder and Habel (2001) and Ahn et al. (2007), for example, both consider the preposition to be part of the temporal expression.

Another rule requires the inclusion in the extent of a temporal expression all syntactic modifiers and postmodifiers, and that the trigger should be the syntactic head.[1] Some example expressions considered in TIMEX2 are presented in Table 2.3.

2.1.3 Summary

In this section, we first presented a general ontology of time-related concepts, consisting of temporal entities (i.e. instants and intervals); situations (also referred to as eventualities), which divide into events, states and processes; and temporal relations. Then, we discussed the notion of a temporal expression and identified two approaches: a broader one, which subsumes all time-related information in a sentence, and a narrow one, associated with work in the domain of information extraction, in which only selected types of time-conveying phrases are taken into account; in our work, we adopt the latter.

2.2 Taxonomising Temporal Expressions

Although any time referred to by a temporal expression has in fact some duration, in our conceptualised view of time in everyday communication some references may be treated as referring to points; these are not ontological durationless instants, but points on a timeline. Of course, we also use temporal expressions to refer to periods of time, in which case we mean to express some duration. In our work we will refer to the times (both points and intervals) referred by temporal expressions as **temporal entities**.

The transfer of the ontological notions of instants and intervals developed in the philosophical literature to the world of processing temporal expressions in texts can be attributed to Mani and Wilson (2000a), who introduced an annotation scheme based on the ISO 8601 standard (ISO, 2004).[2] They considered two types of temporal expressions: **time points** and **intervals**.[3]

TIMEX2, which is currently the most often used scheme for the annotation of temporal expressions in text, considers three types of expressions: points in time, durations and **set** expressions. The first two types are the same as in the account of Mani and Wilson (2000a); the set type is distinguished to cater for expressions that indicate sets of times (e.g. *numerous Saturdays*) and tell how often things happen (e.g. *every hour*). This three-way distinction is followed in the vast majority of work found in the literature on processing temporal expressions, although alternative terminology is sometimes

[1]Although this is what the TIMEX2 guidelines state explicitly, we observe that this rule is not always obeyed; for example, the head of the expression *the middle of August* is *middle*, not *August*.

[2]The ISO 8601 standard distinguishes between an instant being 'a point on the time axis', and a time point being a 'mark attributed to an instant by means of a specified time scale'. In consequence, two distinct instants may be expressed by the same time point. ISO 8601 also defines a time interval as 'part of the time axis limited by two instants'.

[3]Time points and intervals in the work of Mani and Wilson (2000a) correspond to time points and intervals in the ISO 8601 standard. Their notion of a timeline is equivalent to a time scale, not a time axis, in the ISO standard.

used: for example, Han and Lavie (2004) called the main types **coordinate**, **quantity** and **enumeration**, and Hwang and Schubert (1993) and Hwang and Schubert (1994) divide temporal adverbials into **temporal locations** (e.g. *yesterday*), **durations** (e.g. *for a month*), **time spans** (in the sense an activity is accomplished within that period of time, e.g. *in three hours,*), and **repetition** (e.g. *every half hour*).

Point expressions are the most frequently occurring and sets are the rarest; see, for example, the statistics obtained by Lavelli et al. (2005) for Italian. We have not been able to find a comprehensive listing for English; however, partial results found by Han et al. (2006a) and Jang et al. (2004), our private communication with Lisa Ferro from MITRE, and the statistics of the WikiWars corpus confirm that the same appears to hold true for English.

However, temporal expressions can refer to a given entity in many different ways. A reference to a specific date may be completely context independent, e.g. *3rd January 1996*, but, in specific contexts, it is also possible to use *the 3rd*, *tomorrow*, or *the day we got married* to refer to the same date. The different possible ways in which a temporal expression can refer to a temporal entity require a more detailed analysis of the different types of temporal expressions. As the processing (and in particular, the interpretation) of temporal expressions of different types may require different actions to be taken, we are interested in learning what these types are. In fact, almost all existing work on the detailed taxonomisation of temporal expressions has been carried out in the process of building temporal expression taggers. This has, however, some consequences. As it is very hard to provide a truly broad coverage tagger, these taggers usually do not process all types of expressions, and so invariably the published work tends not to discuss those expressions not covered by the tagger in question. Usually the tagger is built for a specific domain in which some types of expressions may be rare, so an occasional error in the output is not of much significance and not worth the additional development cost and effort. It is also the case that different authors use different terminology when presenting their taxonomies. This makes it more difficult to compare their work, especially in terms of the evaluation of the taggers.

In this section we review existing taxonomies of temporal expressions: we present the different types of temporal expressions that have been identified, and the terminology that has been used to describe them.

2.2.1 Point Expressions

A point expression is an expression which denotes a point on a timeline of some granularity, for example at the granularity of a year. Depending on how the expression refers to that point, it belongs to one of the subtypes distinguished below. The terminology used in the literature differs in this regard; Table 2.4 provides a collation of the terminology used and illustrates crossovers of categories considered in the literature.

All accounts seem to agree that there is a class of temporal expressions that do not require any context for their interpretation; for example, *June 2009* or *25/09/2007*. Originally, Hinrichs (1986) referred to these expressions as **complete** or **independent**, but this terminology did not achieve much traction in the community (only Kimura et al. (2007) also uses the term 'complete'). The most popular names for this type of expression are **fully-specified** (Mani and Wilson (2000b), Puşcaşu (2004), Han et al. (2006a)), **explicit** (Saquete et al. (2003), Schilder (2004), Han et al. (2006a)), and **absolute** (Negri and Marseglia (2005), Vicente-Diez et al. (2008)), the latter being a term introduced at MUC-7.[4] A less common term is **fully-qualified**, used by Ahn et al. (2007).

The second category contains expressions which in some way depend on their context, be it the same sentence, the whole document or the time of making the utterance (e.g. the date of publication). These expressions are therefore called by Mani and Wilson (2000b) **context-dependent**; later Mani (2003) also adopted the term **relative**, as used at MUC-7. Saquete et al. (2003) used the term **implicit** (as opposed to explicit), Puşcaşu (2004) used **underspecified** (as opposed to fully-specified), and Schilder (2004) called them **indexical**, because, as he argued, in order to be interpreted they require an additional index time. Mani (2003), in turn, considered indexical expressions to be only those that are interpreted with respect to the time-stamp of the utterance, which most researchers refer to as **deictic**. However, in the linguistics literature (see, for example, (Akmajian et al., 1990) and (Matthews, 2007)) deictic expressions are divided into indexicals and demonstratives; the former are relative to a specific context (with a specific speaker, addressee, location in space, etc.)—for example, *today*—while the role of the latter is to locate a referent in relation to a speaker (an addressee, or some other person etc.) referred to—for example, *that* in *that day.* But it seems that in the literature on processing temporal expressions, the term 'indexical' is used interchangeably with the term 'deictic', and they determine the same type; in particular, work in this area does not refer to demonstrative expressions (the only exception is Mizobuchi et al. (1998) mentioning that their representation[5] does not support expressions containing demonstrative words). Ahn et al. (2007) distinguished **anaphoric** expressions, which are those that are interpreted with respect to some temporal value provided in the utterance, rather than the time-stamp of the utterance. If the reference time is the time-stamp of an event, as in *17 seconds after hearing the sound*, the expression is often referred to as **event-based**; Setzer (2001) referred to such expressions as **complex**.

It is interesting to observe that little attention has been paid to expressions like *the 15th*, *May 21st*, *April*, or *3 p.m.*, which Busemann et al. (1997) referred to as **underspecified**[6] (note the different use of this term here compared to the already mentioned use by Puşcaşu (2004)). Whether these are considered deictic or anaphoric, there is a different action involved in their interpretation compared to the other subtypes of context-dependent expressions: while expressions like *tomorrow* or *2 days later* are interpreted by calculating offsets from a reference date, underspecified expressions need to have the missing pieces of information (the month, year, day and so on) filled in.

Mani and Wilson (2000b) noted that some point expressions may be used in a **non-specific** fashion. For example, *April* in *April is usually wet* does not denote the month in any specific year, but refers to April in any year in a **generic** way. Similarly, the **indefinite** use of *a Tuesday* in a sentence like *The last election took place on a Tuesday* is not intended to refer to any specific date.

Another group of point expressions distinguished by Mani et al. (2001) are those

[4]However, as there was no interpretation task to be performed at MUC-7 and dates and times were to be tagged separately, an expression like *3 p.m.* was also considered absolute and not dependent on any context. Also, Schilder and Habel (2001) included context-dependent expressions like *24.12* (denoting the 24th of December) or *3 p.m.* in their 'explicit' category.

[5]We note the work of Mizobuchi et al. (1998) concerned Japanese, not English.

[6]Busemann et al. (1997) note that even expressions like *Monday, 4-11-1996* can be considered as underspecified, as in their domain of scheduling a meeting a working day is meant, not a date, which in some companies may be 9am–5pm, but it could also be 8am–7pm.

Authors	Types			
Example	*June 2009*	*tomorrow*	*2 days later*	*May 21st*
Smith (1980)	—	deictic	dependent	flexible anchoring
Hinrichs (1986)	complete / independent	deictic	dependent	flexible anchoring
Busemann et al. (1997)	—	deictic	anaphoric	underspecified
MUC-7	absolute	relative		
Mani and Wilson (2000b), Mani (2003)	fully-specified	context-dependent / relative		
		indexical	—	
Saquete et al. (2003)	explicit	implicit		
Puşcaşu (2004)	fully-specified	underspecified		
Schilder (2004)	explicit	indexical		
Negri and Marseglia (2005)	absolute	relative		
Han et al. (2006a)	explicit / fully-specified	deictic	relative	
Ahn et al. (2007)	fully-qualified	deictic	anaphoric	
Kimura et al. (2007)	complete	non-complete		
		abbreviations	—	
Vicente-Diez et al. (2008)	absolute	relative		

Table 2.4: A comparison of taxonomic types for point expressions.

that refer to temporal units with **vague** or **fuzzy** boundaries. These are year seasons (e.g. *summer*), parts of days (e.g. *morning*) and general references to the past, present and future. Notice that references to these temporal units can be of any type of point expression: fully-specified (e.g. *summer of 1999*), deictic (*next winter*), anaphoric (*the following winter*) or underspecified (*summer*). Therefore, in our view, the vagueness of an expression should be seen as a feature, rather than as a separate type in the taxonomy. It is not the expression that is vague; it is the entity referred to that has vague boundaries.

2.2.2 Period Expressions

An interval, unlike a point, has a **duration** and two points associated with it: a start point and an end point. Interval-referring expressions are also referred to as **periods**, for example by Saquete et al. (2003), who define periods as 'all those expressions that give back a time interval or range of dates'. Mizobuchi et al. (1998) and Mani and Wilson (2000a) included **range expressions** like those presented in Example (2.8), but later Mani and Wilson (2000b) and the TIMEX2 scheme annotated these as two separate point expressions.

(2.8) a. from 3 pm to 6 pm

b. from May 1999 to June 1999

Vicente-Diez et al. (2008) notes that, as some expressions denote an interval by specifying the two boundaries, these expressions may also be considered to be absolute or relative, depending on how the two end points are mentioned; for example, Example (2.8a) in their terminology is a relative interval expression, and Example (2.8b) is an absolute interval expression. Nevertheless, they annotate such range expressions as two point temporal expressions, following the TIMEX2 scheme, and distinguish them from the expression of durations (e.g. *two months*).

We note that some authors refer to periods by means of durations, and others define durations by using periods: Schilder (2004) considers durations first and then talks about periods as **anchored durations**; on the other hand, Mani and Wilson (2000b) discuss intervals as a major type of temporal expressions and then identify **unanchored intervals** as durations. Expressions of durations can also be non-specific; for example *a number of years* or *a few months*.

2.2.3 Set Expressions

Set expressions are expressions referring to more than one instance of a temporal entity, be it a point or a period. For example, *every Tuesday in 1999* refers to a number of days, while *the first three days of every month* denotes several periods each of three days in duration.

As shown by some studies, e.g. by Lavelli et al. (2005), expressions of this type are the rarest. This is one of the reasons why these expressions are little studied compared to other types. Another reason is their complex semantics, which is related to three areas: (1) the taxonomisation of these expressions (i.e. identifying subtypes), (2) the representation of their meaning (i.e. what is the best semantic representation), and (3) their proper recognition and interpretation.

Different authors use different terminology to refer to these expressions. The most popular term is 'set' used by Mani and Wilson (2000a) and later used in the TIMEX2 scheme (Ferro et al., 2005), resulting in many adopting the term (e.g. Ahn et al. (2005a), Hacioglu et al. (2005), Negri and Marseglia (2005), Han et al. (2006a) and Vicente-Diez et al. (2008)). The TIMEX2 scheme defines a set-denoting temporal expression as an expression which indicates *how often* something happens.

Han et al. (2006a) support recognition of some set expressions, such as those in Example (2.9).

(2.9) a. Wednesday and Friday
b. 3pm from Wednesday to Friday

They further distinguished **recurrence** (e.g. *every Tuesday*) and **rate** (e.g. *twice on Wednesday*) subtypes; however, their framework was not capable of representing the semantics of these expressions.

Although many works distinguish the set type, there are differences in what counts as a set expression. For example, Han et al. (2006a) considered the expressions in Example (2.9), but in TIMEX2 these are not set expressions; rather, in each case they are considered to be a number of point expressions.

Pan (2007) studied representation formalisms for what he called **temporal aggregates**, but essentially they were what others refer to as sets of times. Niemi and Koskenniemi (2007) in their representation framework for temporal expressions distinguished **parity** expressions (e.g. *even Tuesdays of the month*), **ordinal** expressions (*every second Monday of the year*) and **containment** expressions (*the months with a Friday 13th*). The main distinction made in TIMEX2 is that between **regularly** and **irregularly** recurring times.

2.2.4 Summary

It seems there is a commonly agreed division of temporal expression into three types: point, duration and set. While there is broad agreement about the scope of these three categories, there are differences regarding the classification into subtypes. Moreover, terminology is not used consistently by different authors. The development of the different taxonomisations we found in the literature was generally driven by the need to adopt different processing to the interpretation of expressions of different types.

The main conclusion we draw from this review is that there is still scope for further analysis of the different types of temporal expressions and of what they have in common. The result would be a refined taxonomy oriented towards the construction of a broad-coverage temporal expression tagger. This would also allow for better comparison of people's work, since at the moment without a commonly accepted taxonomy, it is not always possible to reliably compare the results or talk about the distribution of temporal expressions across types in different genres and data sources.

2.3 Representing Temporal Expressions

Any temporal expression tagger needs to use some representation for temporal expressions. There are, broadly, two levels of representation we might consider: the explicit meaning of a temporal expression, i.e. the context-independent semantics of the expression, and the meaning of the expression in context. In this section we review work that is concerned with either level of representation; in selected cases, we also provide additional explanation about how the particular representation is used in the interpretation of temporal expressions.

2.3.1 Temporal Information in Logic and Formal Semantics

Temporal logic is a relatively large and important field in formal logic, and as such has received a lot of attention, resulting in a number of mature frameworks. Although the source of temporal phenomena considered in these works is often natural language, these frameworks are mainly concerned with the role of temporal reasoning in solving problems in artificial intelligence. In this stream of work we classify the work of, for example, Reichenbach (1947), Prior (1957), Lombard (1978), Allen (1981), McDermott (1982) and Vilain et al. (1990). From the most recent work in this area we note the form of propositional logic called Calendar Logic by Ohlbach and Gabbay (1998), and the OWL-Time ontology developed by Hobbs and Pan (2004) and later extended by Pan and Hobbs (2005) and Pan (2007); this ontology is based on first-order predicate logic. However, it is not straightforward to determine whether the representations used in these models can be easily generated by the automated processing of natural language expressions; in fact, we have not found in the literature any work that would confirm that this has been done.

Formal semantics continues this work by analysing temporal expressions in natural language and formally describing their semantics. This includes the work of Bennett and Partee (1978), Kamp (1979), von Stechow (2002) and Evans (2004). However, our interests lie in more practical approaches which can be successfully implemented in a temporal expression tagger.

2.3.2 Representation via Attributes

The most common approach to the representation of temporal expressions is to store information in a collection of attributes. Each attribute carries a specific part of the expression's semantics. Such a representation is particularly easy to implement, for example as a number of variables of simple data types (e.g. integer or string). Negri and Marseglia (2005) represented a temporal expression via the following attributes/-variables: the type of the expression (with two possible values `T-ABS` and `T-REL`, representing the absolute and relative types, respectively), the presence of modifiers (e.g. *early*), the presence of expressions denoting sets of times (e.g. *every*), the presence of clues concerning the temporal location of the anchor of a period (e.g. *later*), the operator ('+', '–' or '=') encoding the direction of interpretation (e.g. *three days ago* was assigned the '–' operator, and *today* was assigned the '=' operator), the quantity applied to the operator (e.g. three), and the temporal unit of the head of the expression (e.g. day). Puşcaşu (2004) used attributes closely resembling those we find in TIMEX2 and TimeML, and also some attributes representing the context of the expression, for example, `verbTense` to encode information about the tense of the verb governing the temporal expression.

Busemann et al. (1997), in their work on systems for the scheduling of meetings, used hierarchical structures encoded as AVMs. The top-level AVM is uniquely identified by a cooperation identifier (`COOP`), and has `RANGE`, `APPOINTMENT` and `DURATION` information slots; an example is presented in Figure 2.4. `RANGE` denotes the temporal boundaries within which a meeting could be held, e.g. 9am on some date. `APPOINTMENT` has a similar structure to `RANGE`, but concerns the interval of the actual meeting (i.e. the interval of the meeting at the agreed time). `DURATION` expresses the duration of the meeting, storing just a number corresponding to the default unit of hours; for general purpose use this could be extended to a constituent structure holding both a quantity and a unit (e.g. hour and minute).

2.3.3 The Temporal Expression Language

Endriss (1998a) developed a Temporal Expression Language (TEL) for the Verbmobil project (Alexandersson et al., 2000); TEL improved over another representation language, Zeit-Gram (Stede et al., 1998), also developed for Verbmobil. The aim was to provide a representation that is close to linguistic surface forms, but already introduces some level of abstraction before it is later transformed into a canonical representation required for carrying out temporal reasoning and analysis for the purpose of the application.

The representation is based on hierarchical typed feature structures. Any expression is represented as an interval, and the top level structure is `interval description` consisting of `date` structures; these in turn were built of `year`, `month`, `day`, and `time`, with the last three having further defined structure. Some attributes could be left empty. To encompass complex temporal expressions, the language also defines a number of functions, e.g. `next`, `fuzzy` and `that`. Below are a few example representations following Endriss (1998b, Sec. A.3):

(2.10) Today is Friday, January twenty second nineteen ninety three.
```
statement:[today,[dow:fri,month:jan,dom:22,year:1993]]
```

(2.11) In the morning at ten.
```
from:[pod:morning,tod:10:0]
```

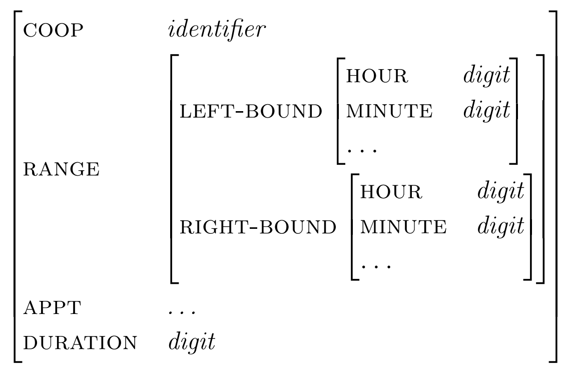


Figure 2.4: The AVM used by Busemann et al (1997) to represent temporal information about a meeting.

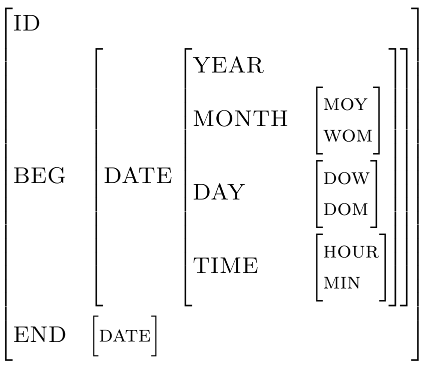


Figure 2.5: A partial view of the 'interval description' feature structure in the framework of Endriss (1998b).

(2.12) Perhaps in the next two weeks.
`during:next(int:dur(2,weeks))`

(2.13) How 'bout the afternoon of Monday the ninth?
`during:[pod:afternoon,dow:mon,dom:9]`

(2.14) Okay, how 'bout Tuesday March the sixteenth sometime after twelve o'clock pm?
`from:[dow:tue,month:mar,dom:16,ex_after([tod:0:0,pod:pm])]`

However, Endriss (1998b, p. 150) acknowledges that in the actual implementation in Prolog the inefficiency related to the representation of underspecified expressions as a set of all possible periods in a given time-frame made it not practical for the project without further improvements.

2.3.4 Temporal Concepts

Mizobuchi et al. (1998) proposed a representation based on 31 **concepts**. A temporal expression is split into **components**, as shown in Figure 2.6a. Each constituent (component) is assigned a concept. For example, *from* is represented by starting (encoded as <`s`>) and *from today* by starting point (<`Ps`>). The full list of concepts is presented in Table 2.5. Some concepts are in a super-sub relation with other concepts; for example starting point (<`Ps`>) is a subtype of point (<`P`>). The concepts are organised into two groups: **object concepts** and **supplement concepts**. Concepts from the first group represent temporal objects, while the concepts from the second group represent supplementary information which can be added to a concept from the first group. For example, a concept starting (<`s`>) can be added to a point (<`P`>) resulting in a starting point (<`Ps`>).

Every component of a temporal expression is represented by a pair, called a **frame**, $[c : v]$, where c is a concept and v is a sequence of **attributes**. The type of the concept determines the type of the frame. For example, `[P: y, m, d]` represents a temporal point with information about year, month and a day, and `[Ps:` p`]` is a starting point which has another point frame p as an attribute. The expression *from 12 November 1997 to 4 June 1998* is represented as `[IP: [P: 1997, 11, 12], [P: 1998, 6, 4]]`.

Two components between which there is a constitutional relationship are merged into a new component for which a conceptual frame is generated by a **merging func-**

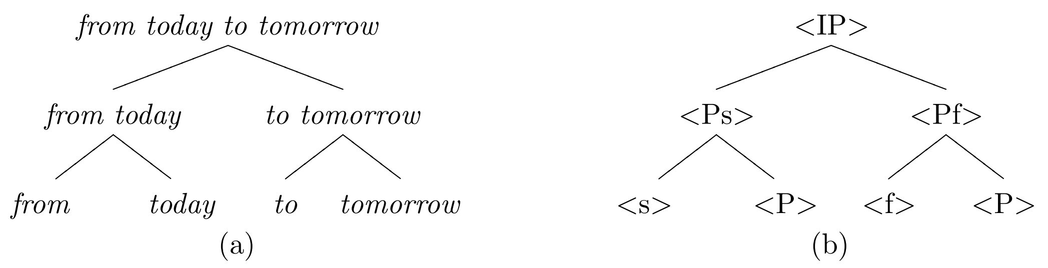


Figure 2.6: An example of expression constituents (a) and their concepts (b) in the formalism of Mizobuchi et al. (1998).

Object Group		Supplement Group	
L (time length)	P (time point)	rf (relative future)	rp (relative past)
Lr (relative L)	Pb (base P)	s (starting)	f (finishing)
Ls (starting L)	Ps (starting P)	bs (base and start.)	bf (base and finish.)
Lf (finishing L)	Pf (finishing P)	b (base)	p (periodic)
Lp (periodic L)	Pbs (base and start. P)	e (extent)	c (connection)
	Pbf (base and finish. P)		
IL (time interv. of L)	IP (time interval of P)		
ILr (relative IL)	IPb (base IP)		
ILs (starting IL)	IPs (starting IP)		
ILf (finishing IL)	IPf (finishing IP)		
PP (periodic P)	PIP (periodic IP)		

Table 2.5: Concepts in the semantic framework of Mizobuchi et al. (1998).

tion. An action table is defined, which for every possible pair of concepts tells whether the concepts can be merged, and if so, what the resulting concept is.

The model also comes with an interpretation algorithm, which for a given string of words generates several possible final semantic frames. Each word is initially assigned all possible frames, which the algorithm then tries to combine with the frames from the already processed part of the expression. The result of the algorithm is a set of possible frame representations for the expression, where from zero to all frames may be correct. The framework and the algorithm were evaluated on a set of 1,995 temporal noun phrases in Japanese. 1,012 expressions (58.1%) were assigned only one frame, which was correct; 848 expressions (34.9%) were assigned several frames, where at least one was correct; and for 135 expressions (7.0%), none of the generated frames was correct.

The model does not handle set (e.g. *every day*), demonstrative (e.g. *in those days*) or non-specific (e.g. *for a long time*) expressions. It also uses only a document time-stamp to interpret underspecified expressions, because no temporal focus tracking mechanism is integrated.

2.3.5 The Time Calculus for Natural Language

Han and Lavie (2003; 2004) developed their Time Calculus for Natural Language (TCNL), which is a typed formal language for encoding the intensional meaning of temporal expressions. This language is built on top of a constraint-based model for human calendars. Each temporal expression is encoded in TCNL, and any underspec-

Expression	TCNL Formula
9th Sep. 1987	$\{1987_{year}\texttt{,sep,}9_{day}\}$
an hour and 30 minutes	$\vert 1_{hour}\texttt{,}30_{min}\vert$
Tuesday and Thursday	$\texttt{[}\{\texttt{tue}\}\texttt{,}\{\texttt{thu}\}\texttt{]}$
Wednesday or Friday	$\{\texttt{wed;fri}\}$
4 o'clock	$\{4_{hour}\vert 16_{hour}\}$
next month	$\{\_ + \vert 1_{month}\vert\}$
exactly one minute ago	$\{\_ - - \vert 1_{min}\vert\}$
the 2nd Sunday in May	$\{\vert 2_{sun}\vert @\{\texttt{may}\}\}$
Wednesday from 3pm to 5pm	$\texttt{[}\{\texttt{wed,}15_{hour}\}\texttt{:}\{17_{hour}\}\texttt{]}$
in the past 3 years	$\texttt{[}\_ : -\vert 3_{year}\vert\texttt{]}$
every 2 minutes and 30 seconds from 3pm to 5pm	$\texttt{[[}\{15_{hour}\}\texttt{:}\{17_{hour}\}\texttt{/}\vert 2_{min}, 30_{sec}\vert\texttt{]]}$
the rest of the year	$\texttt{[}\{\_ + \vert 0_{year}\vert\}\backslash\texttt{[}\{\vert 1_{month}\vert @\{\_ + \vert 0_{year}\vert\}\}\texttt{:}\_\texttt{]]}$
sometime between 3pm and 5pm	$\{(>= 15, <= 17)_{hour}\}$
other than Wednesday	$\{<>\texttt{wed}\}$
less than 1 hour and 30 minutes	$\vert < \vert 1_{hour}, 30_{min}\vert\vert$
sometime before Sept. 9, 1987	$\{\texttt{b}\ \{1987_{year}\texttt{,sep,}9_{day}\}\}$
sometime in 1987	$\{\texttt{i}\ \{1987_{year}\}\}$

Table 2.6: Examples of TCNL formulas.

ification and relativeness are resolved in the interpretation process by the underlying calendar constraint satisfaction system.

TCNL consists of a set of temporal entities and a set of operators and relations. Three types of temporal entities are distinguished: a coordinate (a point in time), a quantity (a duration) and an enumeration (a set of coordinates). Two terms can be connected either by a logical conjunction (','), by logical disjunction (';'), or by linguistic ambiguity ('|'). Examples of temporal expressions and their representations in TCNL are shown in Table 2.6.

The language defines seven operators: three for point-in-time objects, and four for enumerations; '+' is forward fuzzy shifting, '++' is forward exact shifting, '@' is the ordinal operator (selects a point in time with the ordinal specified by the left operand, in the range formed by the right operand), '/' is recurrence, '$\wedge$' is intersection and '\' is a difference operator. Table 2.6 presents a few TCNL formulas using these operators.

Finally, there are three value relations and eight object (entity) relations. The value relations are the standard equal-to ('='), less-than ('<') and greater-than ('>') relations. These relations can be stacked to form single relations, in which case they are interpreted as disjunctive; e.g. ('<>') being not-equal-to. The object relations are those proposed by Allen (1984), which are: before ('b'), meets ('m'), overlaps ('o'), starts ('s'), during ('d'), finishes ('f') and equal ('='). Each of these relations also has an inverse relation. In addition, there is also the in relation ('i') of one interval being wholly contained in another; this is simply a shorthand for (s;d;f), i.e. a disjunction of three relations. See Table 2.6 for some examples of TCNL formulas using relations.

The underscore variable ('_') stands for a reference time (temporal focus) which is used in representations of context-dependent expressions. A temporal focus mechanism in the interpretation process will change it to a specific value depending on the context of the temporal expression.

Once the temporal expressions are encoded in TCNL, they all together describe a temporal constraint-satisfaction problem (TCSP). This consists of a set of constraints over a set of variables; the goal is to find assignments for the variables such that none of the constraints is violated. Providing an answer to a temporal query comes down to adding more variables and solving the augmented TCSP.

2.3.6 A Functional Approach

Usually, the difficulties faced by approaches to semantics when put to use in the field of information extraction are not so much to do with formalisation, but more to do with being robust in the face of real data. On the other hand, approaches to formal semantics are generally restricted to rather limited fragments of grammar, and often do not deal with the kinds of technical issues that implementable systems need to face.

Schilder (2004) proposed a compositional semantics for temporal expressions based on a functional approach, which was meant to provide a link between these two worlds.

The semantics of a temporal expression is represented in a functional way using the lambda calculus. Each individual lexical item that may appear within a temporal expression has a predefined representation; for example, *year* has an entry of `lambda(TS,year(TS))` and *tomorrow* is represented as `lambda(TS,def(next(day(TS))))`, where `TS` is a variable representing some time-stamp (a reference time). To express the context-independent semantics of the whole expression, these representations for individual words are combined together;[7] for example, *the next day* is built from three words with the following lexical entries:

(2.15)
a. `[[the]] =lambda(P, lambda(TS, def(P@TS)))`
b. `[[next]]=lambda(P, lambda(X, next(P@X)))`
c. `[[day]] =lambda(X, day(X)).`

By applying them together in the pattern `the@(next@day)`, we obtain the following representation for the whole expression:

```
(2.16)  lambda(P,lambda(TS,def(P@TS)))@
                  (lambda(P,lambda(X,next(P@X)))@lambda(X,day(X))).
```

Now, by applying β-reduction to this term we obtain a simplified form:

(2.17) `lambda(TS, def(next(day(TS)))).`

Note that this is exactly the same representation as the lexical entry for *tomorrow*; this is as it should be, of course, since the two expressions mean the same thing (it is just the reference time that is different, but this is provided via the `TS` variable).

The semantics of explicit (i.e. context-independent) expressions are derived directly from lexical entries which are in the form `lambda(TS,def(ETS))`, where `ETS` has already some variables bound (e.g. granularity, year, month and day when the expression refers to a date). Durations are represented as `lambda(TS, nr(N, TN(TS)))`, where `N` is the number expressed in the expression (for example, 3 for *3 days*), and `TN` is a function for temporal nouns (e.g. *month* or *year*).

[7] A 'recipe' for how to combine the representations of individual words into a single form is provided by the recognition rule that matches the recognized expression.

The semantics of temporal prepositions are represented in a similar fashion; this allows us to capture the meaning of, for example, *since last month*, given a representation for *last month*. Temporal prepositions are therefore functions which take a specification of time as argument and return a new time.

At the implementation level, each temporal entity is encoded as a list:

(2.18)
```
TS = [WeekDay, date(Year,Month,Day), time(Hour,Minute,Second),
gl(Gran), period(Number,Gran,Quantifier),vague(List)]
```

Here, `gl(gran)` encodes the granularity level, and `vague(List)` lists all modifiers or elements that make the expressions vague, e.g. *beginning* or *summer*. When a point expression is represented, `period()` is not specified; when a period is represented, then `WeekDay`, `date()`, and `time()` are left unfilled.

2.3.7 Timeline Finite-State Transducers

Niemi and Koskenniemi (2007) proposed a model which uses finite-state transducers (FSTs) to mark denoted periods of time on a set of timelines represented as a finite-state automaton (FSA). A temporal expression is first expressed in the **term representation**,[8] which is then converted into a regular expression, which is then further compiled into a **sequence of compositions of FSTs**. These FSTs are then composed with a finite timeline. Such a timeline is represented as a string consisting of nested markings for different calendar periods. For example, a timeline for year 2007 at the month granularity is:

(2.19) [y y2007 [m Jan m] [m Feb m] [m Mar m] . . . [m Nov m] [m Dec m] y].

A calendar expression is represented on a timeline by enclosing the denoted periods of time in marker brackets which have an index corresponding to the expression. All the subsequent representations are shown below for an example expression:

(2.20) **a temporal expression:**
January to March and May 2007
semantic term representation:
intersect(union(interval(month(jan),month(mar)),month(may)),year(y2007))
a sequence of compositions of FSTs:
month(Jan, i1) $\circ$ month(Mar, i2) $\circ$ interval(i1, i2, i3)
$\circ$ month(May, i4) $\circ$ union(i3, i4, i5) $\circ$ year(y2007, i6) $\circ$ intersect(i5, i6, i7)
a finite timeline with the expression marked:
[y y2007 {i7 [m Jan m] [m Feb m] [m Mar m] }i7 [m Apr m]
{i7 [m May m] }i7 [m Jun m] . . . [m Nov m] [m Dec m] y]

At the foundations of the model there are **basic calendar expressions** corresponding to single periods of the Gregorian calendar (e.g. hour, day, month); these also include periods of seasons and holidays, such as Easter or Christmas Day. The term and FST representation of basic expressions may be underspecified; for example *January* is represented as month(Jan) in the term representation and month(Jan, i1) in the FST representation. Apart from basic calendar expressions, the model also contains three other types of expressions: an **interval** (*January to March*), a **list** (*January*

[8]The term representation of Niemi and Koskenniemi (2007) is of course a kind of a functional approach, similar to the approach of Schilder (2004).

8am, except Mondays 9 am	→	`except(h08, mon, h09)`
the following month	→	`nth_following(1, month, then)`
the second Tuesday after Easter	→	`nth_following(2, tue, easter)`
two consecutive Sundays	→	`n_consecutive(2, sun)`
today	→	`containing(day, now)`
three Mondays	→	`any_n(3, mon)`
all Mondays in every May	→	`intersect(mon, may)`
all Mondays in any May	→	`intersect(mon, any_n(1, may))`
any Monday in every May	→	`n_within_each(1, mon, may)`
the courses when the student has free time	→	`impl(course, student_free)`

Figure 2.7: An example use of macros in the framework of Niemi and Koskenniemi (2007).

and May), and **refinement** (*May 2007*). Representations of these types of expressions are obtained by applying operations to the representations of the simpler calendar expressions: interval, union and intersect, respectively. These operations are in fact only macros which resolve to the actual FSTs manipulating the string representations of timelines. For the given example of *January to March and May 2007*, the basic expressions are *January*, *March*, *May* and *2007*. Then, the representation of *January* is combined with the representation of *March* with the operation interval, the result of which is combined with the representation of *May* using union, which is then intersected with the representation of *2007*.

The model also contains other macros; these are: except, nth_following, n_consecutive, containing, any_n, n_within_each, nth_within, and impl. They are used to represent more complex types of expressions (e.g. quantified expressions) and other phenomena; Niemi and Koskenniemi (2008a) describe these macros in more detail; we provide some of their examples in Figure 2.7. FSTs now(i) and then(i) are defined to support deictic and anaphoric expressions (e.g. *today* or *the following month*). There are also additional constructs for expressions like *even Tuesdays of the month* (a parity expression), *every second Monday of the year* (an ordinal expression) and *the months with a Friday 13th* (a containment expression); the coverage of the framework is quite broad.

2.3.8 The Computational Treatment of Temporal Notions

Yet another approach has been developed by Ohlbach (2005) for the Computational Treatment of Temporal Notions (CTTN) system, for which a GeTS[9] functional language has been designed.[10] It is a strongly typed functional language for working with temporal notions. A function computes an interval $[t_1, t_2]$, where t_1 is the start and t_2 is the end of the interval. For example, `tomorrow = partition(now(),day,1,1)` computes the interval starting and ending at a coordinate being +1 from `now()` at the `day` granularity; the expression *this week* is represented in a similar fashion, as shown in Example (2.21a). Example (2.21b) provides a definition that represents a reference to Christmas in a year containing the time point t.

(2.21) a. `this_week(t) = partition(t,week,0,0)`

[9]GeTS stands for GeoTemporal Specification Language.

[10]An extended description of the approach was provided later by Ohlbach (2007).

b.
```
christmas(t) =
    dLet year = date(t,Gregorian_month) in
        [time(year|12|25,Gregorian_month),
         time(year|12|27,Gregorian_month)]
```

However, we are not aware of any temporal expression tagger based on this representation.

2.3.9 Summary

We have presented a survey of frameworks that have been developed for the representation of the semantics of temporal expressions; this concerns both the representation of the meaning of the expressions before and after their interpretation in the context. A precise comparison of the coverage of these solutions is not possible, given the limited descriptions available in the literature. Also, it should be noted that the actual coverage of a tagger based on a specific model may be limited not by the representation itself, but by the state of the development carried out by its authors, as is sometimes explicitly acknowledged.

2.4 Annotating Temporal Expressions

In this section we present, in the order in which they were developed, three schemes used for the annotation of temporal expressions: TIMEX, TIMEX2, and TimeML. Each of these schemes provides annotations as inline SGML markup tags with attributes expressing information concerning the annotated text.

2.4.1 TIMEX

The Message Understanding Conferences (MUCs) were organized as competitions in which participants had to prepare software systems performing a specific task, generally focussed on named entity recognition and event recognition. The conferences introduced a rigorous evaluation paradigm, whereby success or failure was measured in terms of the ability of the systems to replicate human 'gold-standard' annotations within the scope of the task.

At the sixth MUC in 1995, a named entity recognition (NER) task was run for the first time. This task concerned the recognition of references to certain types of entities, such as people, location, and organisations, along with numeric and temporal entities. As part of the annotation scheme developed for this task, occurrences of specific temporal expressions were annotated with the `TIMEX` SGML tag. At the following MUC in 1998, TIMEX was applied to a wider range of expressions, but the design of the scheme remained unchanged. Setzer and Gaizauskas (2000) incorporated TIMEX into a broader scheme which also included annotation of events, states and temporal relations. The most significant change to TIMEX was introduced by Mani and Wilson (2000a), who were interested not only in marking the occurrence of expressions in texts, but also in capturing their meaning. This required extending the annotation scheme to incorporate attributes concerned with semantics.

In the following subsections we present details of the different variants of TIMEX.

2.4.1.1 TIMEX at MUC

In the MUC evaluations, the TIMEX tag had only one attribute specified, `TYPE`, with two possible values: `DATE` or `TIME`. The distinction between expressions of these subtypes was made on the basis of temporal units used in the expression: the TIME subtype was defined as a temporal unit shorter than a full day, whereas the DATE subtype corresponded to temporal units of a full day or longer.

Since MUC-6 was the first evaluative exercise of its kind, the goals set in the task definition (see (Sundheim and Grishman, 1995)) were relatively simple. The scope was restricted to absolute temporal expressions only, which were defined as those which 'indicate a specific segment of time'. In practice this meant that expressions like *the 20th century*, *1st of October*, *midnight*, and *12 o'clock* were to be recognized, but *the last century*, *the first day of the month*, and *morning* were not. Also, in a sentence like *We met in July last year*, only *July* would be annotated, leaving *last year* outside of the annotation. Special days referenced by name, e.g. holidays such as *All Saints' Day*, were not required to be recognized, and were tagged in the gold-standard data as optional. The guidelines also allowed a temporal expression to contain within its textual extent the name of location; this covers expressions of time that incorporate a time zone as, for example, in *1:30 pm Chicago time*.

The NER task definition at MUC-7 (see (Chinchor and Robinson, 1998)) extended the scope of the MUC-6 exercise to include certain relative temporal expressions; these were defined as expressions whose interpretation was dependent on the document time-stamp (e.g. *today*) or which referred to subparts of a temporal unit (e.g. *morning*). In consequence, expressions like the following were to be recognized: *next year*, *last two months*, *10 years ago*, *yesterday*, and *early this year*. Adjacent expressions of distinct types were to be separately annotated; combined with the extension of the scope, this meant that expressions like *yesterday evening* resulted in two TIMEX tags. The only exceptions to this rule were cases where marking all items separately would be confusing, as in *4:15 p.m. Tuesday local time*, where the rule would lead to the pattern TIME DATE TIME.

Some aspects of the MUC-6 guidelines that had been left underdetermined were now tightened up; for example, determiners that introduce temporal expressions and words or phrases modifying the expressions, such as *around* and *about*, were explicitly not to be tagged. So, for example, *shortly after the 4th of May* was to be annotated in the following way: `shortly after the <TIMEX TYPE="DATE">4th of May</TIMEX>`. However, expressions like *the end of 1991* or *late Tuesday* were not considered as modified expressions, but as expressions containing both relative and absolute elements, and therefore were annotated as single TIMEXs. Indefinite and vague date expressions, unanchored durations and event-based expressions were also not to be tagged; this included, for example, *now*, *recently*, *for the past few years*, and *since the beginning of arms control negotiations*. The underlying rule was that the values denoted by the annotated expressions must be markable on a timeline.

Holidays referenced by name were now compulsorily tagged. It was also decided that dates from alternate calendars, such as fiscal years, the Hebrew calendar, Julian dates, and even 'Star Dates' from the fictional Star Trek universe were to be considered markable temporal expressions.

2.4.1.2 Extensions to TIMEX

Setzer's TIMEX Setzer and Gaizauskas (2000) proposed a scheme for annotation of events, time expressions and temporal relations. The part concerning temporal expressions did not really differ from the TIMEX scheme used at MUC-7, both in terms of the design and range of expressions that the scheme was used for. The only difference was the addition of a `TID` attribute, which uniquely identified each temporal expression in a document for the purpose of associating it with events.

Later, Setzer (2001, pp. 103–106) added a `calDate` attribute, which expressed the date represented by the expression. The value of the attribute was in the format `[[DD]MM]YYYY` or {`‘SPR’|‘SUM’|‘AUT’|‘WIN’`}`YYYY`. For expressions describing times related to events, Setzer (2001) further extended the structure of the `TIMEX` tag. For example, the expression *17 seconds after hearing the sound* was considered to be **complex** and was annotated as follows:

(2.22) `<timex tid=5 type=complex eid=3 signalID=7 relType=after>`17 seconds `</timex> <signal sid=7>`after`</signal> <event eid=3>`hearing`</event>` the sound

The `type` attribute indicated what further attributes should be used; `eid` stored the ID of the event associated with the time expression; `signalID` stored the ID of the annotation of the string describing the temporal relation between the temporal entity and the event; and `relType` specified the type of that relation. As we can see, there was no attribute representing the type of the temporal expression or its temporal value.

Mani and Wilson's TIMEX The most significant change in the design of TIMEX was introduced by Mani and Wilson (2000a), who added new attributes representing the meaning of expressions: `VAL`, `VAL1`, `OFFSET` and `DIRECTION`. The first two attributes were used to encode points on a calendric timeline using the ISO 8601:1997 format: `CC:YY:MM:DD:HH:XX:SS`.[11] In the case of point-referring expressions, only `VAL` obtained a value; for period expressions the two attributes encoded the end points of the period. For example, *from 3pm to 6pm* was annotated as one temporal expression of type TIME, with `VAL` referring to 3pm and `VAL1` referring to 6pm.

The scheme assumed interpretation not only of those expressions which were relative to the document time-stamp, as was the case in MUC-7, but also those expressions whose reference times were expressed elsewhere in the body of the text. For those expressions which could not be interpreted because their reference time remained unknown, the `OFFSET` and `DIRECTION` attributes were to be used. The value of the former was specified in the ISO 8601 format, expressing the number and granularity of temporal units that the date or time was distant from the reference time; the latter indicated the direction of the offset simply by using the `+` or `-` character. For example, *2 months later* would be encoded as `OFFSET="00:00:02" DIRECTION="+"`.

The scheme, however, still did not encompass a wide range of expressions; those

[11] In the `CC:YY:MM:DD:HH:XX:SS` format `CC` encodes a century, `YY` encodes a year within that century, `MM` stands for a month, and `DD` for a day; `HH:XX:SS` encodes information about time, i.e. hour, minutes and seconds. For example, the value `19:80:04:21:14` encodes the date of the 21st April 1980 and the time 9:14pm with no seconds specified. See `http://www.iso.org/iso/support/faqs/faqs_widely_used_standards/widely_used_standards_other/date_and_time_format.htm` for more information about the ISO 8601 standard.

presented in Example (2.23) were not supposed to be interpreted, and those in Example (2.24) were not to be recognized at all.[12]

(2.23)	a.	*April* is usually wet.	generic
	b.	I was born on *a Tuesday.*	indefinite
	c.	I had swimming classes on *every Tuesday in 1999.*	gapped interval
	d.	*Winter 1999* was extremely severe.	vague
	e.	We got married *about three years ago.*	approximate
(2.24)	a.	*The first three days of every month* are always the busiest.	set of intervals
	b.	The movie is *two hours long.*	unanchored interval
	c.	She left *five days after he came back.*	event-dependant

Not much later, Mani and Wilson (2000b) presented a modified scheme in which the `VAL` attribute could also have a value in the week-based format, `CC:YY:Wwwd`, where `ww` is the ISO week number, and `d` is the ISO day number in the week. The expression *last week* would then take the value `20:00:W16` if it referred to the sixteenth week of year 2000. The format of the attribute value was also extended to represent some temporal units not handled in the ISO 8601 standard. For example, year seasons were encoded as `CC:YY:aa`, where `aa` is a code for a year season; the expression *summer of 1969* was annotated as `VAL="19:69:SU"`. The `VAL1` attribute was dropped; in consequence, the string *from 3pm to 6pm* was processed as containing two temporal expressions, *3pm* and *6pm*. The `OFFSET` and `DIRECTION` attributes were also removed.

2.4.2 TIMEX2

The TIMEX annotations are insufficient for real-life text processing applications such as information extraction, question answering and summarization, which require more than just recognition that certain sequences of tokens correspond to a temporal expression. If we want to carry out any kind of temporal analysis, we need to have the expressions dereferenced or interpreted so that, for example, we know exactly what days are meant by expressions like *two weeks after Christmas* or *five days later.* Moreover, a variety of different conventions are used to format absolute time expressions: the date referred to as *16th of May, 2000* can also be expressed as *16/05/2000*, *05/16/2000* or *2000/05/16*, amongst a variety of other formats. For many tasks, clearly some form of normalisation is required, so that temporal entities can be compared and placed into sequence. Finally, even a normalized date in the sense just described may not be enough. If we are dealing with documents sourced in different time zones, or which refer to times in different geographical regions, we may have to normalise times and dates to some common standard; this requires identifying the local time zone and taking account of any local summer time conventions, for example.

The TIMEX annotation format used in the MUC competitions was not able to express this kind of information. Mani and Wilson (2000a) made a step forward by introducing some degree of interpretation of the meaning of temporal expressions, but it was only the TIMEX2 scheme that provided a level of detail useful enough to become an informal standard in the research community.

[12]The labels used in these examples adopt the terminology of Mani and Wilson (2000a).

TIMEX2 was initially developed at MITRE in 2000 under the Translingual Information Detection, Extraction, and Summarization (TIDES) program. It has subsequently been used in many other projects, including the Time Expression Recognition and Normalisation (TERN) task of the Automatic Content Extraction (ACE) evaluations. Although there are still a number of contentious areas in its design, TIMEX2 is now a robust standard that is expressively adequate for many practical applications.

2.4.2.1 The Design of TIMEX2

A detailed description of the usage of the TIMEX2 tag (i.e., what kinds of expressions it provides annotations for, how the values of the attributes should be determined, and what the extent of the annotated text should be) is beyond the scope of the presentation here, since it occupies over 50 pages of guidelines; by comparison, the specifications of TIMEX for the MUC-6 and MUC-7 evaluations fitted into 2.5 and 5 pages, respectively.

TIMEX2 defines a temporal expression as an expression which 'tells us when something happened, or how long something lasted, or how often something occurs' (Ferro et al., 2005, p. 6). This is then limited to expressions containing chosen trigger words, and further narrowed down with rules concerning the use of trigger words and determining the extent of temporal expressions.

As well as marking the occurrence of temporal expressions in texts, TIMEX2 also attempts to represent the meaning of these expressions. This is a big step forward compared to the earlier TIMEX scheme; however, it introduces a number of challenges. The most recent version of TIMEX2 (Ferro et al., 2005) (also considered to be the final version) represents the semantics of temporal expressions by means of five attributes: `VAL`, `ANCHOR_VAL`, `ANCHOR_DIR`, `MOD`, `SET`. These attributes are used, respectively, to encode the temporal location of a point on a timeline or the duration of a period; to encode the temporal location of one of the period's end-points; to capture the direction of temporal reference from the anchoring point; to express modifications to more basic temporal values; and to flag whether the temporal expression refers to a set of temporal entities. TIMEX2 also allows annotators to place any remarks in a `COMMENT` attribute. A summary of the TIMEX2 attributes and the values they take is provided in Table 2.7.

The values of the `VAL` and `ANCHOR_VAL` attributes are the most diverse of all the attributes. They use a string representation based on formats defined in the ISO 8601:2004 standard: calendar date (`YYYY-MM-DD`), week date (`YYYY-Www-D`), time of day (`hh:mm:ss`), date and time (`YYYY-MM-DDThh:mm:ss`), and duration (`PnYnMnDTnHnMnS` or `PnW`). TIMEX2 extends the encodings provided in the ISO standard by introducing tokens representing additional temporal granularities: for example, in place of a month number, TIMEX2 also permits codes for year seasons (e.g. `SU` for summer), half-years (e.g. `H1`), and quarter-years (e.g. `Q3`). It also added support for BCE[13] years, references to the distant past (i.e. billion, million, thousand years ago) and general references to past, present and future. For non-specific use of expressions (as in, for example, *a sunny day in June*) TIMEX2 uses an uppercase `X` to fill the slots at the unspecified granularities (here, `XXXX-06-XX`). In regard to the encoding of duration, TIMEX2 adds new temporal units for decades, centuries and millennia. This significantly extended the range of values of the `VAL` attribute compared to the scheme of Mani and Wilson (2000a) and substantially increased the coverage of the scheme, making it more suitable for the annotation of real data.

[13]BCE stands for 'before current era'.

Attribute Name	Description
VAL	Contains a normalized form of the date, time or duration. The value is in the ISO 8601-compliant format or in ISO-extended format.
MOD	Captures temporal modifiers, using values BEFORE, AFTER, ON_OR_BEFORE, ON_OR_AFTER, LESS_THAN, MORE_THAN, EQUAL_OR_LESS, EQUAL_OR_MORE, START, MID, END and APPROX.
ANCHOR_VAL	Contains a normalized form of an anchoring date or time. The value is in the ISO 8601-compliant format or in ISO-extended format.
ANCHOR_DIR	Captures the relative direction or orientation between VAL and ANCHOR_VAL attributes, as in WITHIN, STARTING, ENDING, BEFORE, AFTER and AS_OF. It is used to express the information about when a duration (including general references to past, present and future) is placed.
SET	Identifies expressions denoting sets of times; either takes the value YES or is empty.
COMMENT	Contains any comment that the annotator wants to add to the annotation; ignored from the point of view of automatic processing of the text.

Table 2.7: The attributes of a TIMEX2 tag.

In contrast to TIMEX, embedded annotations are possible in TIMEX2. These occur in three situations: time-anchored expressions (e.g. [*two weeks from* [*next Tuesday*]]), possessive constructions (e.g. [[*this year*]*'s summer*]), and when the pre-modifier and the head are both triggers (e.g. [*the* [*June 2004*] *period*]).

2.4.2.2 Older Versions

There were also two earlier versions of TIMEX2. The differences between the most recent version and the intermediate version from 2003 are either not significant or concern very rare temporal expressions, so from a practical point of view they have little influence on the annotation of real data; details of the changes are listed by Ferro et al. (2005, pp. 5, 8).

However, the changes from the 2001 version of TIMEX2 are more significant. This includes the set of attributes used to represent the meaning of annotated expressions. The 2001 version did not yet include the ANCHOR_VAL and ANCHOR_DIR attributes, but instead had PERIODICITY, GRANULARITY and NON_SPECIFIC attributes.

PERIODICITY and GRANULARITY were used to annotate sets of recurring times; PERIODICITY was used only for regularly recurring times (i.e. when the recurrence is expressed by words like *always*, *every* and *each* or when periodicity is given explicitly with words like *daily*, *monthly* and *yearly*). The value of PERIODICITY was in the format F*nu*, where *n* is an integer or X, and *u* encoded a temporal unit of the cycle (e.g. D for day, M for month). For example, for both *every month* and *monthly* the value of the attribute was F1M. GRANULARITY represented the unit of time denoted by each entity of the set. The value was in the same format as the value of PERIODICITY, except preceded by a G. Example (2.25) presents a reference to a regularly recurring time, and Example (2.26) presents an expression referring to a set of times recurring irregularly.

(2.25)
```
<TIMEX2 VAL=1999-WXX-2 SET=YES GRANULARITY=G1D PERIODICITY=F1W>
every Tuesday in <TIMEX2 VAL="1999">1999</TIMEX2></TIMEX2>
```

(2.26) `<TIMEX2 VAL=1998-WXX-6 SET=YES GRANULARITY=G1D>numerous Saturdays </TIMEX2>`

The `NON_SPECIFIC` attribute explicitly indicated whether an expression was non-specific, in which case the attribute had value `YES`; otherwise the value was simply empty. The non-empty value was applied to both generic statements (e.g. *I love December*) and indefinite references (e.g. *The election took place on a Tuesday*). The uppercase `X` placeholder in the `VAL` attribute was also used.

2.4.2.3 Shortcomings

A lot of time and effort was spent on the development of TIMEX2, incorporating comments from ACE participants who tested the scheme in practice on a large scale. However, there are still a few areas which are imprecise, debateable, or simply under-developed.

The guidelines distinguish between the annotation of so-called geological eras and specific years before our era. Without context, it is not clear whether the expression *8 thousand years ago* should be annotated as a reference to a geological era (`VAL=KA8`) or as a specific year (`VAL=BC5992`, assuming the reference year is 2008). Of course, in a sentence like *The species X died out 8 thousand years ago* it is not the particular year that is meant; rather, this is a reference to a certain era in a geological sense. However, the example provided in the guidelines (Ferro et al., 2005, p. 15) *The king lived 4,000 years ago* is not necessarily referring to a specific year either; however, in the guidelines it is annotated as referring to the year 2001 BC, on the assumption that the utterance is made in 1999. What is lacking here is a clear distinction between the specific versus ‘ballpark’ character of temporal references.

Another problem concerns the annotation of the anchor of a period when that anchor is a point whose representation uses the `MOD` attribute; for example, *the following seven months* may be anchored to a point referred to by the expression *mid-September*. The TIMEX2 guidelines stipulate that the `MOD` attribute is used in conjunction with the `VAL` attribute (Ferro et al., 2005, Sec. 4.4). Consequently, in the given example, the semantics of *-mid* is lost. One solution to this problem would be the addition of a new attribute `ANCHOR_MOD`.

Another issue related to the `MOD` attribute is that although it is designed to be applied to references made to parts of units, such as their beginnings, middles and ends, other types of references to parts cannot be represented; for example, for *most of 1956* there is no value that can be used to indicate the fact that it is not the whole of 1956 that is referred to.

One of the rules in TIMEX2 is that the leading preposition should not be included in the extent of an annotated expression, despite the fact that the preposition is often used in determining the semantic value of the expression. For example, *fiscal 1998* is treated as a point when used in the phrase *in fiscal 1998*, but as a period if it appears in *throughout fiscal 1998* (Ferro et al., 2005, p. 31).

It also seems somewhat counterintuitive that a string like *late on Friday* is considered to be one temporal expression (Ferro et al., 2005, p. 62), but *8:00 p.m. on Friday* is annotated as two temporal expressions (Ferro et al., 2005, p. 63). Here the guidelines would be more consistent if in both cases there was just one annotation.

There are also some examples in the guidelines that in our opinion should not be treated as temporal expressions. For instance, TIMEX2 attempts to capture the semantics of rate expressions, such as *per hour* in *The Orient Express traveled at speeds of 50 miles per hour* (Ferro et al., 2005, p. 41). Not only does the scheme provide a misleading annotation (the proposed annotation `VAL=PT1H, SET=YES` suggests an unspecified number of repeating one-hour-long time chunks), arguably the expression does not refer to *any* set of points or periods in time. It is not the case that something is happening n times each hour; the expression of speed just indicates what distance can be travelled within a period of one hour. As we note elsewhere, the representation of expressions referring to sets of temporal entities is generally underdeveloped in TIMEX2.

Although the specification of TIMEX2 claims to support time zones (Ferro et al., 2005, p. 17), there are also limitations in this regard. First of all, what one finds added to the value of the `VAL` attribute is a time difference (e.g. `+10`), not a code corresponding to a time zone (e.g. `AEST`); the time difference encodes both the time zone and the daylight saving adjustment of one hour. Secondly, while it is said that the ISO 8601 standard is adhered to—which also allows for minutes to appear in the format of a time difference—the guidelines explicitly limit the use of time differences to only those that are full hours; this is problematic for handling some regions, for example, South Australia lies in +`09:30`.

TIMEX2 is oriented to the northern hemisphere by assuming winter ends in March (Ferro et al., 2005, p. 30). Without any marker in the scheme indicating the hemisphere, it is either not clear how to interpret a value `YYYY-WI` found in an annotated text; or, following the assumption made in TIMEX2, it must refer to the first quarter of the year, which is incorrect for processing documents published in the southern hemisphere. This is a real obstacle to universal application of the scheme everywhere around the world without further adjustments.

TIMEX2 leaves its users the possibility of making extensions to suit the needs of particular applications: for example, more precise support for time zones, or the interpretation of holidays and event-based expressions (by default, no dates are assigned in these cases). This, however, creates problems for the comparative evaluation of taggers, given that particular systems may have implemented different approaches in these areas.

2.4.3 TIMEX3 in TimeML

TimeML is an annotation scheme for events, temporal expressions and temporal relations. The roots of the scheme can be seen in the work done by Setzer and Gaizauskas (2000), who proposed the inline SGML-based annotation tags `event`, `timex` and `tr_signal` (the specification of the `timex` tag here was borrowed from the MUC-7 evaluations). Later, Setzer (2001) modified her scheme to a version which we have already described in Section 2.4.1.2. Also in 2001, the first public version of TIMEX2 was published. TimeML was then developed, based on the results of Setzer (2001) and TIMEX2, in the context of the six-month workshop on Temporal and Event Recognition for Question Answering Systems (TERQAS) as part of the ARDA-funded AQUAINT program; the report presenting the outcomes of the workshop was prepared by Pustejovsky et al. (2002), and constitutes the definition of TimeML version 0.1.0. Since then, TIMEX2 and TimeML have been developed concurrently and independently, and even the most recent specification of TimeML (Boguraev et al., 2005; Saurí et al., 2006) still refers to TIMEX2 from 2001.

expressions in the appositives:
```
<TIMEX2>the 1960s, <TIMEX2>the days of free love</TIMEX2></TIMEX2>
<TIMEX3>the 1960s</TIMEX3>, <TIMEX3>the days of free love</TIMEX3>
```
conjoined expressions:
```
<TIMEX2>six months</TIMEX2> or <TIMEX2>a year from <TIMEX2>now</TIMEX2></TIMEX2>
<TIMEX3>six months</TIMEX3> or <TIMEX3>a year</TIMEX3> from <TIMEX3>now</TIMEX3>
```
event-anchored expressions:
```
<TIMEX2>five days after he came back</TIMEX2>
<TIMEX3>five days</TIMEX3> after he <EVENT>came</EVENT> back
```
nested expressions:
```
<TIMEX2>two weeks from <TIMEX2>next Tuesday</TIMEX2></TIMEX2>
<TIMEX3>two weeks</TIMEX3> from <TIMEX3>next Tuesday</TIMEX3>

<TIMEX2><TIMEX2>This year</TIMEX2>'s summer</TIMEX2>
<TIMEX3>This year's summer</TIMEX3>

<TIMEX2>the <TIMEX2>June 2004</TIMEX2> period</TIMEX2>
the <TIMEX3>June 2004</TIMEX3> period
```

Figure 2.8: A comparison of extent annotation in TIMEX2 and TimeML.

Since version 0.1.0, the TimeML scheme has consisted of seven semantics-related SGML tags (`EVENT`, `MAKEINSTANCE`, `TIMEX3`, `SIGNAL`, `TLINK`, `SLINK` and `ALINK`), but their application has changed over time. The full specification of TimeML is available from the project's website.[14] In the context of our work, we are most interested in the `TIMEX3` tag, which is used to annotate temporal expressions.

2.4.3.1 Annotation of The Extent

TIMEX3 is applied to the same range of expressions as TIMEX2 as specified by Ferro et al. (2001) and Ferro (2001), but in several cases there are differences in marking the textual extent of annotations. Compared to the most recent version of TIMEX2, the differences concern expressions appearing in appositive constructions, conjoined expressions, event-anchored expressions and cases where TIMEX2 assumed nested annotations (in TimeML nesting is not allowed); example expressions and their annotations in both TIMEX2 and TimeML are presented in Figure 2.8. TIMEX3 also introduced **empty tags**, also referred to as **empty content tags** or **abstract tags**, which had no extent in the text; consider Example (2.27).

(2.27)
```
John begins teaching <TIMEX3 tid=t1 type=DURATION value=P1W
beginPoint=t2 endPoint=t3>one week</TIMEX3> from
<TIMEX3 tid=t2 type=DATE value=XXXX-9-15>September 15</TIMEX3>.
<TIMEX3 tid=t3 type=DATE value=XXXX-9-22 temporalFunction=TRUE
anchorTimeID=t1/>
```

The last annotation (`tid=t3`) does not span over any text; it is only used to provide the temporal value corresponding to the ending anchor of the period referred to by the expression *one week*.

[14]See `http://timeml.org`.

Attribute Name	Description
`tid`	Contains the ID of the TIMEX3 annotation in a document. The value is in the format of 't' followed by an integer, for example `t14`.
`type`	Stores the type of the annotated expression. There are four possible values: `DATE`, `TIME`, `DURATION` and `SET`.
`functionInDocument`	Specifies the role the expression plays in a document. The possible values are `CREATION_TIME`, `MODIFICATION_TIME`, `PUBLICATION_TIME`, `RELEASE_TIME`, `RECEPTION_TIME`, `EXPIRATION_TIME`, or `NONE` (which is equivalent to no value).
`value`	Stores the value of the expression after the full interpretation process. The attribute gets a value in the ISO 8601 or ISO-extended format, just as `VAL` does in TIMEX2.
`mod`	Indicates that the value of the expression is augmented in some way. The usage of this attribute is the same as in TIMEX2.
`temporalFunction`	Indicates whether the value of the expression is determined by an external function or not; i.e. whether the expression is context-dependent or context-independent. The possible values are `true` and `false` (the latter also corresponds to an empty attribute value).
`valueFromFunction`	Used only when `temporalFunction` is `true`. Stores a reference to a temporal function that can provide the value of the expression. In the process of manual annotation this attribute is ignored.
`anchorTimeID`	Used only when `temporalFunction` is `true`. Contains the ID of the expression that serves as the reference time in the interpretation of the expression in question.
`beginPoint`	Determines the beginning anchor of a period by storing the ID of another `TIMEX3` annotation, which specifies a point in time.
`endPoint`	Determines the ending anchor of a period by storing the ID of another `TIMEX3` annotation, which specifies a point in time.
`quant`	Defined as any text (i.e. there are no predefined values for this attribute) and generally it is a literal from the annotated text that quantifies over the expression; e.g. `EVERY`. Used only when the type of the expression is `SET`.
`freq`	Contains an integer and a time granularity marker (such as `D` for day, `M` for month, `X` if not specified) to represent any frequency expressed; e.g. `2X` for *twice*. Used only when the type is `SET`.

Table 2.8: The attributes of a `TIMEX3` tag.

2.4.3.2 The Semantic Representation

The specification of the TIMEX3 tag defines 12 attributes that can be used to represent the meaning of temporal expressions; they are summarized in Table 2.8. Two attributes are sourced directly from TIMEX2: `value` and `mod`. Compared to TIMEX2, the new features are: specification of the expression's type (although only capturing top-level distinctions); reference to a temporal function that can interpret context-dependent expressions; reference to the temporal expression that functions as a reference time; provision of two attributes specifying the anchoring points of a period; provision of two attributes capturing the semantics of expressions; referring to sets of temporal entities; and specification of the special role a temporal expression may play in a document.

Some of the changes in the approach to annotating the extent of expressions influence the way the expressions are interpreted. For example, *two weeks from next Tuesday* in Example (2.28) would be interpreted in TIMEX2 as a specific date with a nested expression *next Tuesday* which is a point.

(2.28) I'm leaving on vacation two weeks from next Tuesday.

In TIMEX3 this string is processed as containing two separate temporal expressions: *two weeks* being a duration, and *next Tuesday* being a point. Additionally, the end point of the period would be provided as an abstract tag. A similar difference occurs for event-anchored expressions like *five days after he came back* in Example (2.29).

(2.29) We saw him five days after he came back.

In TIMEX2 this expression could either be left uninterpreted or provided with a point-referring representation; in TimeML only *five days* is tagged as a temporal expression and its value is specified as a duration of five days.

Finally, we note that although TimeML made some improvements to the representation of set-referring temporal expressions, the solution it provides is still not ideal. For example, the expression *the past three summers* is represented by `type=SET value=XXXX-SU freq=3Y`, but the representation would not be any different for the expression *the future three summers.*

2.4.4 Summary

In this section we reviewed three annotation schemes used for marking temporal expressions in texts: TIMEX, TIMEX2 and TimeML.

Although all the three schemes are meant to be used for the annotation of temporal expressions, there are considerable differences among them; not only in their design (see Table 2.9 in which we list the attributes of the `TIMEXx` tag in the different schemes), but also in the range of the expressions they can be applied to, in the approach to which text tokens are marked up, and in the semantic representations they provide.

At the present time, the most widely-used annotation scheme is TIMEX2. This is a result of, amongst other things, its use in the TERN task at three ACE evaluations (in 2004, 2005 and 2007), which required the preparation of a number of annotated corpora and forced the participants to adjust their taggers to this scheme. It provides a reasonable coverage of the phenomena concerning the recognition and interpretation of temporal expressions; in comparison, TIMEX, although used at two MUC evaluations, is just too simplistic for any real task of natural language processing. On the other hand, TimeML provides a richly expressive language for capturing information about time; it goes much beyond the scope of TIMEX2 by integrating the annotation of temporal expressions with annotation of events and temporal relations. The range of phenomena covered in TimeML allows for the development of applications that require a sophisticated analysis of the temporal relations in text, such as the visualisation of car accidents on the basis of textual reports (see (Berglund et al., 2006)) or question answering (see the report prepared by Radev and Sundheim (2002)). However, there are very few publicly available resources that support this scheme and no large scale evaluations have been organized.

Annotation Scheme	Attributes
TIMEX (MUC-6)	TYPE, STATUS
TIMEX (MUC-7)	TYPE
TIMEX (Setzer and Gaizauskas, 2000)	TYPE, TID
TIMEX (Setzer, 2001)	TYPE, TID, (calDate | (EID, signalID, relType))
TIMEX (Mani and Wilson, 2000a)	TYPE, VAL, VAL1, OFFSET, DIRECTION
TIMEX (Mani and Wilson, 2000b)	TYPE, VAL
TIMEX2 (Ferro et al., 2001)	VAL, MOD, SET, PERIODICITY, GRANULARITY, NON_SPECIFIC, COMMENT
TIMEX2 (Ferro et al., 2004)	VAL, MOD, SET, ANCHOR_VAL, ANCHOR_DIR, NON_SPECIFIC, COMMENT
TIMEX2 (Ferro et al., 2005)	VAL, MOD, SET, ANCHOR_VAL, ANCHOR_DIR
TimeML (Pustejovsky et al., 2002)	VAL, MOD
TimeML (Saurí et al., 2006)	type, tid, functionInDocument, anchorTimeID, value, valueFromFunction, temporalFunction, mod, beginPoint, endPoint, quant, freq, comment

Table 2.9: The attributes of the TIMEX annotation tag in various annotation schemes.

2.5 Corpora with Annotated Temporal Expressions

In the area of natural language processing, an annotated corpus is a very useful resource for conducting research. Such a corpus can be used in two ways: as data illustrating some research problems for which one would like to develop algorithms and solutions, and as benchmarking data for the evaluation of competitive algorithms and methods.

We are interested in studying the identification and interpretation of temporal expressions in texts, and so corpora annotated with temporal expressions are of great importance to us. Now that we have presented the various schemes available for the annotation of temporal expressions, in this section we describe the available corpora related to this research, indicating their accessibility and content.

2.5.1 The MUC Corpora

As discussed above, the evaluations at the 6th and 7th Message Understanding Conferences (MUCs) included a named entity task; one of the entity types distinguished was the time expression, or **timex** for short. The corpora used at these evaluations are publicly distributed by the Linguistic Data Consortium (LDC): the MUC-6 corpus has catalogue number LDC2003T13,[15] and the MUC-7 corpus is marked as LDC2001T02.[16] Both corpora contain news articles. The MUC-6 documents are sourced from the Wall Street Journal (years 1987, 1989, 1993, 1994), and the MUC-7 documents are from the New York Times news service from the period from 1st January to 11th September 1996. In each case, the corpus consists of three parts: training (development) data, dry run test, and final evaluations.

In Table 2.10 we summarize a quantitive analysis of the two datasets. In total, there

[15]See http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2003T13.

[16]See http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2001T02.

	Development				Dry Run Test				Final Evaluation			
Corpus	Docs	TIMEX (total)			Docs	TIMEX (total)			Docs	TIMEX (total)		
		date	time	other		date	time	other		date	time	other
MUC-6	318	1531			30	122			30	117		
		1524	7	0		120	2	0		117	0	0
MUC-7	100	1441			100	1361			100	1481		
		1173	267	1		1133	228	0		1258	222	1

Table 2.10: Statistics for the MUC-6 and MUC-7 corpora.

are 1770 TIMEX annotations in the MUC-6 data, and 4283 annotations in the MUC-7 data. While the size (in terms of the number of annotations) of the development components is comparable, the evaluation data used for MUC-7 is about twelve times larger than for MUC-6. We also notice that while in MUC-6 data temporal expressions of the TIME type practically do not occur (only nine instances in total), in MUC-7 the ratio of time to date expressions is much higher and time expressions constitute about 20.1% of all temporal expressions. One reason for this is that, in the MUC-6 data, document time-stamps are just dates (e.g. *02/03/93*), but in MUC-7 data the time-stamps are in the form of a date followed by time (e.g. *01-03-96 1442EST*). Because in the MUC evaluations an annotation cannot involve taggable expressions of two distinct subtypes (i.e. a date followed by a time or a time followed by a date must be annotated separately), the document time-stamps in the MUC-7 corpus provide 300 TIME expressions, accounting for about 41.8% of all annotations of this type.

We also note that in two cases in the MUC-7 corpus an annotator assigned the `DATE|TIME` string to the `TYPE` attribute of a TIMEX annotation, probably because the annotator could not decide whether the expression referred to a date or a time. Such a value is not allowed by the MUC-7 annotation guidelines (see, (Chinchor and Robinson, 1998)) and is considered a mistake in the data. The confusing expressions for the annotator were *More than 36 hours after the crash* and *early Tuesday morning*. These cases are counted as 'other' in Table 2.10.

2.5.2 The TIDES Parallel Temporal Corpus

In Section 2.4.2 we already mentioned the TIDES program funded by DARPA. Apart from the design of the TIMEX2 scheme, another deliverable from this program was the TIDES Parallel Temporal Corpus. Its preparation consisted of translating to English and aligning the Spanish part of the Enthusiast corpus of dialogs,[17] and annotating temporal expressions in the English version with the TIMEX2 tags as specified in the TIDES annotation guidelines version 1.0.2 (Ferro et al., 2001).

The original dialogs in Spanish were obtained by asking subjects to set up a meeting based on partially filled calendars they were given; the conversation were then transcribed. The translation to English was performed by a commercial human translation service, providing high quality translations. The annotation of temporal expressions in the English version of the transcriptions was carried out by six students from George-

[17]The Enthusiast corpus was prepared under the JANUS project funded by the US Department of Defence (DOD). This project was focused on speech-to-speech machine translation.

	ACE 2002			ACE 2003			ACE 2004		
Domain	Docs	Words	TIMEX	Docs	Words	TIMEX	Docs	Words	TIMEX
Broadcast News	85	17,922	628	147	34,681	1,050	222	61,621	1,848
Newspaper	17	14,682	337	–	–	–	–	–	–
Newswire	78	34,134	926	102	58,592	1,547	116	58,543	1,711
Arabic Trans.	–	–	–	–	–	–	58	13,466	526
Chinese Trans.	–	–	–	–	–	–	37	12,522	365
Total	180	66,738	1,891	249	93,273	2,597	433	146,152	4,450

Table 2.11: Statistics for the ACE 2004 Time Normalization Training Corpus.

town University.

The size of the corpus is 95 documents in SGML format. Each document contains a single dialog, and each speaker's turn is marked with one SGML element (`<spanish>` or `<english>`). The corpus is parallel and organised such that the English translation of an element immediately follows the Spanish original. In total, the corpus contains 7305 temporal expressions, 3541 of which (48.47%) are in the English part of the corpus. These statistics have been obtained after fixing a number of problems with incorrect `TIMEX2` tags, a list of which we present in Appendix B. The original corpus is publicly available to download in the Annotated Corpora section of MITRE's webpage on TIMEX2 (see `http://timex2.mitre.org`).

2.5.3 The ACE 2004 Corpora

The introduction of the Time Expression Recognition and Normalization (TERN) task at the Automatic Content Extraction (ACE) program in 2004 obviously required relevant training and evaluation corpora to be prepared. The training corpus is now distributed by LDC,[18] but the evaluation corpus has not been released to the general public.

The training corpus consists of documents used at the ACE 2002, 2003 and 2004 evaluations, but now all annotated with TIMEX2 (2003 v.1.3). All three subsets contain news data; additionally, the ACE 2004 subset contains English translations from the Chinese Treebank English Parallel Text Corpus and from the MT-2003 translation dataset of the Arabic Treebank 1 Corpus. In total, the corpus contains 862 documents with 8938 TIMEX2 annotations; Table 2.11 presents in detail the number of documents in the different subparts of the corpus. Each document exists in two versions: one unannotated and one with inline gold-standard `TIMEX2` annotations.

We note that various documents in the corpus have different SGML structures. In particular, document time-stamps are placed in different parts of documents: this is significant because correct identification of the time-stamp is important for automatic temporal tagging; very often its value can be used as a reference time for interpreting temporal expressions found in the document.

[18]See `http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2005T07`; the full name of the corpus is *ACE 2004 Time Normalization (TERN) English Training Data v 1.0*. For tasks other than TERN, there was a separate corpus prepared and named *ACE 2004 Multilingual Training Corpus* (LDC2005T09); it did not have temporal expressions annotated. The source data of the TERN corpus largely overlap with the English source data contained in the Multilingual Training Corpus.

```
<timex2 ID="CNN_CF_20030303.1900.00-T9" VAL="P2Y"
        ANCHOR_VAL="2003" ANCHOR_DIR="STARTING">
  <timex2_mention ID="CNN_CF_20030303.1900.00-T9-1">
    <extent>
      <charseq START="2996" END="3013">the next two years</charseq>
    </extent>
  </timex2_mention>
</timex2>
```

Figure 2.9: An example stand-off annotation for a temporal expression in the XML APF file format.

2.5.4 The ACE 2005 Corpora

For the ACE 2005 evaluations a new set of corpora was prepared. As in the case of the ACE 2004 corpora, the ACE 2005 Training corpus is publicly available via the LDC,[19] but the evaluation corpus is not.

The training corpus for the TERN task consists of the complete set of English, Arabic and Chinese training data used for the 2005 Automatic Content Extraction (ACE) evaluations.[20] Documents in Arabic and Chinese have temporal expressions annotated only at the detection level, while annotations of documents in English also provide the results of the interpretation task. However, as our work concerns only English, we will not analyse here the content of the datasets prepared for other languages.

Temporal expressions are annotated with the TIMEX2 scheme (v. April, 2005). Unlike corpora prepared for the earlier ACE evaluations, this corpus was annotated with stand-off annotations listed in the XML APF file format;[21] in Figure 2.9 an example annotation of the temporal expression *the next two years* is presented. The corpus has four annotated versions of each document: two with annotations of extent made by two independent annotators, another which was created by a third annotator based on the first two versions, and finally a version with meaning representation (the TIMEX2 attributes) added.

This corpus turns out to be the largest available corpus with temporal expressions annotated. In Table 2.12 we present some statistics concerning the size of the corpus, showing also the split of documents across the domains.[22] In Table 2.13 we present statistics concerning a modified version of the corpus, which constitutes the dataset we will use in our work; this version of the corpus was created by removing six documents due to the `&` character not being correctly encoded as XML entities and, in consequence, making the gold-standard annotations corrupted.

[19] See `http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2006T06`.

[20] Note that the Arabic and Chinese collections here contain documents written actually in Arabic and Chinese, not translations from these languages into English as was the case with the ACE 2004 TERN Training Corpus. We also note that although the corpus contained data for Arabic, the ACE 2005 evaluations were not carried out for Arabic.

[21] APF stands for the *ACE Program Format*.

[22] The domain codes denote as follows: BC for Broadcast Conversations, BN for Broadcast News, CTS for Telephone Conversations, NW for Newswire, UN for UseNet Newsgroups, and WL for Weblogs.

Domain	Docs	Words	TIMEX2
BC	60	40,415	626
BN	226	55,967	1,455
CTS	39	39,845	409
NW	106	48,399	1,235
UN	49	37,366	741
WL	119	37,897	1,003
Total	599	259,889	5,469

Table 2.12: Statistics for the ACE 2005 Training Corpus.

Domain	Docs	Data size [b]	TIMEX2
BC	59	260,413	618
BN	225	378,874	1,450
CTS	39	299,139	409
NW	106	333,328	1,235
UN	49	235,415	741
WL	115	249,251	975
Total	593	1,756,420	5,428

Table 2.13: Statistics for the Modified ACE 2005 Training Corpus.

Domain	Docs	Data size [b]	TIMEX2
BC	9	48,722	142
BN	74	75,731	322
CTS	6	54,522	70
NW	106	209,973	894
UN	13	48,377	167
WL	46	137,549	433
Total	254	574,874	2,028

Table 2.14: Statistics for the ACE 2007 Evaluation Corpus.

Domain	Docs	Data size [b]	Words	TIMEX2
BC	9	48,722	7,499	142
BN	74	75,731	10,049	322
CTS	6	54,522	7,531	70
NW	34	71,492	10,410	305
UN	13	60,669	7,503	167
WL	19	47,371	7,299	148
Total	155	358,507	50,291	1,154

Table 2.15: Statistics for the ACE 2005 Evaluation Corpus (v2.0).

2.5.5 The ACE 2007 Corpora

For the ACE 2007 evaluations, there was no new training corpus released for the TERN task in English. Participants could use the ACE 2005 Training and Evaluation corpora to develop their tagging systems. For the final evaluations, NIST developed a corpus containing 254 documents in the same six domains that were used in ACE 2005 evaluations. This corpus was not prepared from scratch, but by extending the ACE 2005 Evaluation Corpus: 72 documents were added in the newswire domain and 27 documents extended the weblog subset.[23]

Details of the corpus are presented in Table 2.14, and in Table 2.15 we show statistics for the underlying ACE 2005 Evaluation corpus. We note that there are almost twice as many gold-standard annotations of temporal expressions in the 2007 corpus than in the 2005 version, which makes the 2007 edition a better choice to use for evaluations. However, from our experience with working with this corpus we can say there are many annotation errors, which significantly blurs the real performance of an automatic tagger. We also note that the corpus is not publicly available.

[23]Private communication with NIST revealed that the ACE 2005 Evaluation Corpus was released to participants by mistake; to keep the evaluations fair, it remained up to the participants not to use the ACE 2005 evaluation data for development and training of their systems.

2.5.6 The TimeBank Corpus

The TimeBank 1.2 Corpus[24] contains 183 news articles annotated with TimeML 1.2.1,[25] which results in 1414 TIMEX3 annotations of temporal expressions. The official statistics say the corpus contains 61,000 non-punctuation tokens.

Since we use TIMEX2 in the work presented here, this corpus is of limited practical use for us: we cannot use it for evaluation, but we can inspect it to see what temporal expressions it contains. We also note that it seems that the annotators did not follow exactly the TimeML 1.2.1 specifications (Boguraev et al., 2005) and annotation guidelines (Saurí et al., 2006). For example, in the whole corpus there is not a single occurrence of an empty TIMEX3 tag, which, as described in the documentation of the annotation scheme, should be used for specification of one of the end points of an anchored duration. This should be taken into account when evaluating any automatic taggers on this corpus.

2.5.7 Summary

In this section we have presented a number of corpora of English texts that are annotated with temporal expressions, indicating their size in terms of the number of documents and annotations, the annotation schemes used, any issues related to using the corpus in practice, the genre of the documents, and the availability to the general public.

Of the presented corpora, the MUC corpora, the TIDES Parallel Temporal Corpus, the ACE 2004 and 2005 training corpora and TimeBank (both versions 1.1 and 1.2) are publicly distributed. None of the ACE evaluation corpora (2004, 2005 and 2007) is publicly available, but they were released to participants of the ACE evaluations in particular years. As the ACE 2007 evaluation corpus has been created by extending the 2005 evaluation corpus (99 documents were added), it is more reasonable to evaluate automatic temporal taggers on the newer corpus.

The MUC corpora, annotated with TIMEX, reflect a simplistic approach to what constitutes a temporal expression and are focused on recognition, not interpretation; therefore their usefulness is quite limited for our research, although their content can serve as an additional dataset to investigate the range of expressions that can be found in real documents in the news genre.

The TIDES program initiated research on the interpretation of temporal expressions and resulted in the development of the TIMEX2 scheme and a relevant corpus containing translations of transcripts of conversations about scheduling meetings. For the purpose of running the subsequent ACE evaluations, at which more mature versions of TIMEX2 were used, new annotated corpora were prepared. Given their size and the fact that they contain documents from six genres, they have become de facto standard benchmark data in this area, despite numerous errors we have found in the annotations.

TimeBank is the newest corpus of all that we reviewed; its distinctive feature lies in the annotation scheme used, TimeML, which encompasses the annotation of events and temporal relations, which in turn opens the door to more research areas. However, the complexity of TimeML, and the differences to TIMEX2 regarding the annotation

[24]See `http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2006T08`.

[25]For each document there is also an extra file with additional annotations of, for example, sentences and named entities.

of temporal expressions, have prevented TimeML from surpassing the popularity of TIMEX2 yet. Given these circumstances, our choice is to use TIMEX2, not TimeML.

Many of the issues related to using the existing and available corpora are due to limitations of the annotation schemes used to annotate the data; for example, the annotations do not encode the taxonomical types of expressions (except the very high level distinction between point and period, or date and time) or what temporal reference times should be used to interpret the expressions. The most common domain of texts found in the existing corpora is news, with a few other domains present in the ACE corpora. However, long narratives with complex discourse structure are definitely underrepresented; in particular, the existing datasets can often be correctly interpreted by using the document time-stamp as the reference time, thus not encouraging research into more complex narrative structures where the reference time must be appropriately selected from the temporal expressions mentioned earlier in the text. On top of this, we found there are a reasonable number of annotation errors in the data. All these issues should be taken into account when using the corpora for evaluation of automatic taggers.

2.6 Approaches to the Processing of Temporal Expressions

At least a limited-in-scope temporal expression recognition functionality is implemented in many computer systems, as dates are common and at the same time very important types of expressions. Explicit dates in a few formats (for example, *2001-07-18* and *18/07/2001*) can be detected and parsed into constituents with a relatively small set of regular expressions available in many popular programming languages like Perl, Python or Java. However, development of a broad-coverage and large-scale temporal expression tagger is a much more challenging task because of the numerous other forms of temporal expressions, which makes recognition difficult, and the context-dependent cases that require a sophisticated analysis of the surrounding text to derive the full meaning.

Attention to the development of taggers for temporal expressions began in the research community with the 6th MUC evaluations in 1995. However, at that time, the range of target expressions was limited and no meaning analysis was required. With the appearance of the TIMEX2 annotation scheme (see Section 2.4.2), the area has attracted more interest, and the subsequent ACE evaluations have significantly boosted the number of new systems developed; in Table 2.16 we collate some basic information on the tagging systems that we have found presented in the literature.

Generally, there are two approaches that can be used in the development of a tagger: the rule-based approach and the machine-learning approach. The former requires more effort in the phase of building recognition grammars and interpretation rules, while the latter needs some reasonable amount of training data. The results reported in the literature show that rule-based approaches usually give better accuracy, probably due to the scope for precise knowledge engineering by means of meticulous rule writing; on the other hand, the machine-learning-based tools can be more easily adapted to new domains and languages.

System Name	Authors	Languages	Recognition	Interpretation
Cosma	Busemann et al. (1997)	German	rule-based	rule-based
—	Mizobuchi et al. (1998)	Japanese	rule-based	rule-based
TempEx	Mani and Wilson (2000b)	English	rule-based	rule-based+ML
—	Vazov (2001)	French	rule-based	—
TagTime	Baldwin (2002)	French, English	rule-learning	rule-learning
KTX	Jang et al. (2004)	Korean	rote-learning	rule-based
—	Berglund (2004)	Swedish	rule-based	rule-based
—	Schilder (2004)	English, German	rule-based	rule-based
—	Ahn et al. (2005a)	English	rule-based	rule-based
—	Ahn et al. (2005a)	English	ML-based	rule-based
CTEMP	Wu et al. (2005b)	Chinese	rule-based	rule-based
—	Wu et al. (2005a)	Chinese	—	multi-class.+rules
—	Vichayakitti and Jaruskulchai (2005b)	Thai	rule-based	rule-based
ATEL	Hacioglu et al. (2005)	English, Chinese	ML-based	—
Chronos	Negri and Marseglia (2005)	English, Italian	rule-based	rule-based
TERSEO	Saquete (2005), Saquete et al. (2007)	Spanish, English, Italian, Catalan	rule-based + translation	rule-based
GUTime	Verhagen et al. (2005)	English	rule-based	rule-based
—	Treumuth (2006), Treumuth (2008)	Estonian	rule-based	rule-based
—	Bethard and Martin (2006)	English	ML-based	—
TEA	Han et al. (2006a)	English	—	rule-based
TimexTag	Ahn et al. (2007)	English	ML-based	rule-based+ML
—	Saue (2007)	Estonian	rule-based	—
—	Kimura et al. (2007)	English, Japanese	rule-based	rule-based
—	Vicente-Diez et al. (2008)	Spanish	rule-based	rule-based
—	Parent et al. (2008)	French	rule-based	rule-based
—	Bassara (2008)	Polish	rule-based	rule-based
—	Bittar (2009)	French	rule-based	rule-based
TETI	Caselli et al. (2009)	Italian	rule-based	—
TERNIP	Northwood (2010)	English	rule-based	rule-based
HeidelTime	Strötgen and Gertz (2010)	English, German	rule-based	rule-based

Table 2.16: Existing temporal expression taggers. The differences between rule-based and machine-learning (ML) approaches, and the explanation of rote-learning and multi-classification, are provided in the text.

2.6.1 The Architecture of Taggers

It is common that taggers have two distinguished modules: a recogniser and an interpreter. The recogniser finds occurrences of temporal expressions in texts, possibly storing some information about the semantics of the expression's constituents, and the interpreter provides the final value of the expression. For example, consider the following sentence:

(2.30) I saw John *two days earlier* in the shopping center.

When processing this, the recogniser would determine that there is a temporal expression starting with the token *two* and ending on *earlier*, that the granularity of the expression is in terms of days and that it expresses an offset of two days to the past of some reference date. The interpreter would then select the reference date from the context and subtract two days from it to get the final date as the temporal value of the expression.

In its simplest form, the recognition grammar may be just a flat collection of rules, resembling a set of regular expressions. The specific implementation environment imposes constraints on the design of the grammar, such as the order of rules, their priorities in matching a string, or the possibility of reusing one rule in another. An example of a recognizer having all its rules in a single component is that implemented by Saquete et al. (2002).

Recognition may be divided into steps, possibly dividing the recognition module into submodules of specific functionality. For example, Chronos (Negri and Marseglia, 2005) has two sets of rules: basic rules and composition rules. The latter are applied only after the former. Basic rules are used for detection (i.e. finding triggers, like *second* or *April*; at this point this may often generate many spurious matches), bracketing of temporal expressions (i.e. establishing the correct extents of expressions in text) and gathering contextual information (such as the presence of any modifiers) that will be used later in the interpretation stage. The set of composition rules is used to resolve conflicts, which arise when basic rules produce overlapping or adjacent annotations. The task of composition rules is to choose one out of multiple possible taggings, or to combine these into one annotation. For example, when processing a sentence containing the string *the whole Monday night*, one basic rule may recognize only *the whole Monday* and another rule may recognize *Monday night*; a composition rule combines the two extents into a single one. Similarly, in their tagger for Chinese, Wu et al. (2005b) apply constraint rules after basic rules to reject some matches which, based on some heuristics, are not considered to be true temporal expressions. For example, in Chinese *very* is represented in the same way as *ten+minute*.

The processing path inside the interpreter is usually split, so that depending on the type of the expression (for example, explicit point-referring, deictic point-referring, and duration) different actions are taken. Some system descriptions mention an additional module for discourse processing, which is used in the interpretation phase by a mechanism for determining a reference time for context-dependent expressions. In practice very simplistic heuristics are implemented based on a recency model[26] (e.g. (Mani and Wilson, 2000b)) or the module is left not implemented at all (e.g. (Puşcaşu, 2004)).

A tagger may also have additional modules for pre- and post-processing (e.g. outputting results in a specified format, like TIMEX2) as in the systems developed by Vicente-Diez et al. (2008) and Northwood (2010). The benefit of such a modular design is improved extensibility of the tagger, for example to provide different output formats. However, Northwood (2010) notes that if the target annotation schemes (or rather their underlying taxonomical distinctions) are quite different then it is very difficult to implement a tagger with an output-independent internal representation of temporal expressions.

[26]In Section 6.3.2 we survey in more detail various methods used for temporal focus tracking.

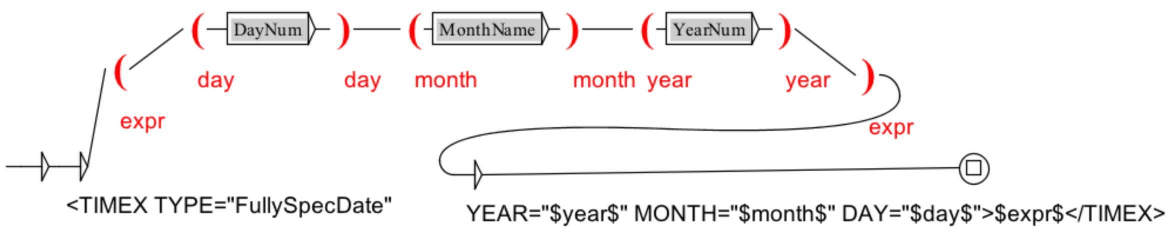

DayNum
MonthName
YearNum
day
day
month
month
year
year
expr
expr
<TIMEX TYPE="FullySpecDate"
YEAR="$year$" MONTH="$month$" DAY="day">$expr$</TIMEX>

Figure 2.10: A finite-state transducer (FST) recognizing a fully-specified date in `DayNum MonthName YearNum` format and generating a `TIMEX` tag around the matched expression with attributes expressing the local semantics of the expression.

2.6.2 Rule-based Systems

The so-called rule-based systems are those in which the processing is driven by a set of rules which encode the required knowledge. The underlying formalism is the finite-state transducer (FST).

FSTs are very useful for linguistic applications and have been successfully applied to many tasks, including text tokenization, morphological analysis and shallow parsing (see, for example, Karttunen (2001) for more details). Temporal expression tagging can also be carried out by means of FSTs; Figure 2.10 presents an example FST which we built in Unitex.[27] This FST recognizes a fully-specified date in the format `DayNum MonthName YearNum`, for example *18 May 2005*, and adds to the original string an inline `TIMEX` annotation with some date-referring attributes by specifying the output on selected arcs. The graph refers to other graphs, `DayNum`, `MonthName` and `YearNum`, which recognize constituents of the expressions and possibly generate some output; these subgraphs are also FSTs. We thus have an example grammar consisting of four FSTs.

2.6.2.1 Basic Temporal Expression Tagging

Even basic capabilities of temporal expression recognition can be very useful for a specific application. For example, Orphanides (2008) in his work on textual entailment extracted only fully-specified date-referring temporal expressions of year, month and day granularity and date ranges in selected formats (for example, *August 1 to September 2, 2003*; *August 1-2, 2003*). Year identification was carried out with a regular expression matching all numbers between 1000 and 2100.

Berglund (2004, Ch. 6) developed a module as a part of a system creating 3D animations of car accidents based on police reports in Swedish.[28] Because of the limited domain, the module did not need to recognize some types of expressions often considered by other authors; the out-of-scope types included periods (*for six hours*), parts of years (*summer of 1992*, *the Easter holiday of this year*), and temporal expressions appearing as parts of noun phrases (*Yesterday's accident*). Also any expression that would require a dedicated rule in their development environment, e.g. a fixed multi-word phrase such as *St. Knut's Day*,[29] was not recognized. The date-referring expres-

[27]See `http://www-igm.univ-mlv.fr/~unitex`.
[28]See (Berglund et al., 2006) for a description of the project.
[29]*St. Knut's Day* is a traditional festival celebrated in Sweden on 13th January.

sions that were in the scope of the application were interpreted using the document time-stamp as the reference; time-referring expressions (i.e. those with no indication of a date, e.g. *a quarter to eight*) were not interpreted with respect to any date.

A slightly more sophisticated approach to the interpretation of the recognized expressions was employed by Kimura et al. (2007), who constructed a system for the automatic collection of information about people from English and Japanese webpages, resulting in the generation of a timeline of events involving these people. The time recognition module constructed for this purpose processed only calendar dates, which could be fully-specified or context-dependent; other types, such as periods (*five days*), sets (*every Tuesday*), or fuzzy expressions (*summer*, *early part of October*, *some years later*) were not handled. Each of the recognized expressions was then interpreted with one of three rules, depending on the type of the expression: underspecified (*May 23*), coreferential (*the same month*), or relative (*yesterday*). The reference time used in the interpretation process was the temporal value of the most recent temporal expression.

We are of course interested in a wider range of expressions than the coverage of each of the above systems.

2.6.2.2 Grammar Size

The size of the recognition grammar, measured by the number of rules, differs across different systems. Of course, since the rules can vary significantly in complexity, a comparison based on such a measure is not completely meaningful, but it seems to be the only reasonable and easy-to-report quantitative description of grammars. The number of rules can range from around sixty (in TempEx[30]) or a hundred (Vazov, 2001) to over 1,000 (Negri and Marseglia, 2005) or even 1,400 (Treumuth, 2006). The size of the grammar can also be reported as the number of states and arcs in the FST. Schilder and Habel's (2001; 2003) system tagged expressions with an FST consisting of 15 states and 61 arcs, while Vicente-Diez et al. (2008) used 25 states (12 of which were final) with 20 predicates on arcs specifying types of elements to accept next as a constituent of the expression.

The size of a grammar is related not only to its coverage (although there are no principles for the relationship between the size and coverage), but also to the granularity of rules. It is desirable to have an efficiently implemented grammar, but it should also be easy to read, understand and extend. While one would not want to have one rule per one possible string to be recognised, a small number of very complex rules is also not necessarily a sign of a well-designed grammar. If rules become too universal or match semantically different types of expressions, it may be very difficult to provide semantic values for the expressions or to augment the grammar in future.

2.6.2.3 Structure of Rules

Rules are usually in the `PATTERN` ⇒ `ACTION` form, where the `PATTERN` specifies what the rule is supposed to match in text, and the `ACTION` contains instructions on what and how to annotate. The actual implementation of such rules may look different depending on the development environment. For example, Mani and Wilson (2000a) implemented their TempEx in Perl, and the grammar consists of a number of text substitutions based on regular expressions in the following manner: `$documentText`

[30]As found in the implementation available at `http://timex2.mitre.org/taggers/timex2_taggers.html`.

`=~ s/(PATTERN)/<TIMEX2>$1<\/TIMEX2>/g`. Here, the rule substitutes every occurrence (the `g` switch for global) of the pattern in the text of the document with the string specified between the two slashes following the pattern. The substitution can refer to parenthesised parts of the pattern by means of numeric variables; as a result the rule produces an annotation with the text matched with the pattern inserted between the opening and closing TIMEX2 tags.

Bittar (2009) implemented his recognition grammar in Unitex, which provides a GUI interface for drawing FSTs as graphs, in the manner we presented in Figure 2.10. Their representation is then compiled directly into FSTs; the knowledge engineer does not work with rules of the above or any similar format at all.

Schilder et al. (2003) implemented their system in Prolog. Their rules have the form `HEAD` ⇒ `BODY`, where the `HEAD` part specifies the result of the rule (i.e. what has been recognized), and the `BODY` part describes what has to be matched; this can refer to tokens and other already recognized elements, or can call other rules. Example (2.31) presents two rules which recognize context-dependent expressions.[31]

(2.31) a. `indexical(ID, Val) ---> @tempAdv(ID, Val).`

b. `indexical(ID, DetSem@(ModSem@NounSem)) --->`
`@det(DetSem), @mod(ModSem), @tempNoun(ID, NounSem).`

The rule (2.31a) recognizes temporal adverbials (for example, *tomorrow*), and the rule (2.31b) recognizes expressions of the `Determiner Modifier TemporalNoun` pattern (for example, *the next year*); `indexical(ID, DetSem@(ModSem@NounSem))` specifies that the functional representation of the recognized expressions is obtained by applying `DetSem` to the result of applying `ModSem` to `NounSem`.

2.6.2.4 Information Used in Rules

Rules in all the systems we are aware of use the lexical surface form of tokens as the basis for recognizing the majority of temporal expressions. Given the relatively large number of lexical units to be considered by taggers, lists and dictionaries are commonly used to keep month, weekday and holiday names, temporal units, trigger words and spelled-out numbers (for example, *fourteenth*).

Part-of-speech tags are often used, for example by Mani and Wilson (2000b), Schilder (2004), Ahn et al. (2005a) and Negri and Marseglia (2005), to improve the recognition coverage and precision by providing a means to write both more general and more specific rules. An example of the first would involve substituting the matching a number of specific lexical items with a single POS tag; and an example of the latter would involve matching a given lexical item only when it is used in a chosen role, for example, *second* being a noun, but not a verb or ordinal number. These generalisations also make the development of a grammar easier, providing a way of decomposing the ruleset. POS tags help to identify numbers, prepositions and determiners, which are common constituents of temporal expressions; alternatively, the rules would have to list all possible tokens (for example, *a*, *the*, and *this* for determiners) or contain subpatterns, for example, to match numbers. Adjectives and nouns can be used as modifiers (for example, as in *a warm night*), and should be included in the extents of temporal expressions, but of course enumerating in a rule all possible lexical forms in such cases is not possible; using a POS tag is simply necessary here. POS tags can also be employed to find verbs to disambiguate some cases by rejecting ambiguous lexical forms which do not constitute temporal expressions, as in Example (2.32).

[31]In Schilder's terminology these are called indexical expressions.

(2.32) a. *May* I have ice-cream now?

b. Soldiers started to *march* immediately.

c. I don't think Bush's second term will *last* four years.

Identification of verbs can also help in the interpretation stage; the tense of a verb is used, for example, by Mani and Wilson (2000b) and Han et al. (2006a) to decide whether a bare weekday name should be interpreted as a date earlier or later than the reference date.

The use of syntactic information seems to be quite limited. Ahn et al. (2005a) used a chunk parser to determine the extent of an expression detected with a trigger word. However, due to errors introduced by the chunk parser, this version of the tagger scored worse than when the extent was obtained with rules using POS tags.

In order to provide correct temporal values, the interpretation rules need, of course, to implement the calendar and basic operations on it, such as addition of a number of days to a date. This is also necessary if the system is supposed to verify the semantic correctness of input data; for example, if a processed utterance is *I would like to meet you on Monday Nov. 2 1996*, the system could realise that 02-11-1996 was not a Monday.[32] Such additional functionality is not a common feature in the existing taggers, and Cosma, a rule-based system developed by Busemann et al. (1997) as a German language interface to meeting scheduling agent systems, is the only system we found in the literature that performed such a verification step.

2.6.2.5 Interpretation

The way in which temporal expressions are interpreted, i.e. how the value of the referred-to temporal entity is determined, depends on the representation used for the semantics of the expression.

In many cases, for example in Chronos (Negri and Marseglia, 2005), the meaning of expressions without context is represented by means of a number of variables which store various kinds of information about the expression. Then in the interpretation stage, a further set of rules uses the reference time and information stored in these variables to derive the final representation of the referred-to entity.

Things are done a little differently by Schilder (2004). Here the representation of meaning is in a functional form: for example, *the next year* is represented as `lambda(TS, def(next(year(TS))))`, where `TS` is a reference time. To find a value for this term, there is a predicate `computeTimex`, which for each function found in the representation of the expressions (for example, `next()` or `year()`) has a rule specifying how to modify the argument (an encoding of a temporal entity).

An original and interesting approach was developed by Han et al. (2006a). The interpretation process is seen as a constraint satisfaction problem (CSP). Once the expression has its TCNL representation generated by FSTs, the interpretation is carried out by the CSP solver. The reasoning is implemented with an interval-based AC-3 algorithm (see (Ruttkay, 1998)) with a chronological backtracking mechanism.

2.6.2.6 Development Time

One of the arguments raised in the literature against rule-based systems, for example by Ahn et al. (2007), is that the crafting of grammars is time-consuming. Negri and

[32] 02-11-1996 was a Saturday.

Marseglia (2005) reported that the development of 1000 basic rules took one person-month. Although this did not constitute the whole system, and an annotated corpus was used to build the rules (which simplifies the task), it does suggest that building a system with satisfactory performance is possible in a reasonable amount of time (construction of the basic rules was the most laborious part of the task). In the absence of a training corpus, it would still be possible to develop a system based on intuitions; but a corpus is essential for machine-learning based approaches, and the time and cost of corpus preparation are not to be underestimated. We should also note that if a new version of a rule-based system is to be developed for another domain (or even for a new language), the experience would most likely help to shorten the construction time compared to the first system one develops.

2.6.2.7 Portability to New Languages

An issue often raised as a disadvantage of rule-based systems in general, compared to the machine-learning approach, is the effort required to adapt the system to a new language or domain. However, in the area of temporal expression recognition, some existing research suggests that transformation to other languages does not necessarily mean writing new grammars from scratch.

Saquete et al. (2006) experimented with automatic translation of rules from Spanish to English, and then from both Spanish and English to Italian; later, another transformation was carried out from Spanish to Catalan (see (Saquete et al., 2007)). The results suggest that although a completely automatic process cannot achieve high accuracy, it can be very helpful in establishing a first version of a rule-based system for a new language. Continuing the development manually from that point instead of from scratch can reduce the amount of time for porting a system to a new language. This should be especially the case for pairs of similar languages like Spanish and Catalan or Spanish and Italian. It remains to be determined how well such approaches would work for pairs of languages from different families: for example, English (Indo-European) and Arabic (Afro-Asiatic).

Wilson et al. (2001) also note that the recognition and interpretation processes are very similar across many languages. The differences concern the different sets of lexical triggers: some can be directly translated, but there are also cases which do not have counterparts in other languages. We can find such examples for Polish vs. English; for example, *pojutrze* is *the day after tomorrow* and *przedwczoraj* means *the day before yesterday*. The interpretation process looks to be more language-independent than the recognition stage, because it mainly involves the application of calendar arithmetics, which is related to the calendar used in the given community rather than a language.

2.6.3 Machine-learning-based Systems

Just as for the rule-based systems, some of the first systems built to detect dates using a machine-learning approach were developed for the MUC evaluations. An example is BBN IdentiFinder (see, (Bikel et al., 1999)), which uses Hidden Markov Models. However, given the limited range of temporal expressions considered in the MUC evaluations (see Section 2.4.1.1) we will not analyse the results of this work. The basic idea was that as dates were seen as a specific type of named entity, systems developed for named entity recognition were trained to recognize one additional class of entities; no interpretation was carried out by these systems.

Later work included attempts to use semi-automatic rule writing, and using machine-learned classifiers for recognition of temporal expressions as considered in the TIMEX2 scheme. There has also been some potential identified for machine learning in the interpretation step.

2.6.3.1 Semi-Automatic Rule Writing

Before we present work using the typical machine-learning paradigm for the recognition and interpretation of temporal expressions, we first report on some attempts at facilitating rule development using a semi-automatic process.

The first attempt to eliminate the manual crafting of recognition and interpretation grammars was made, to the best of our knowledge, by Baldwin (2002), who implemented an example-based learner. The idea was to use annotated examples to learn the dependencies between lexical items found in the extent of temporal expressions and values in TIMEX2 (v2001) attributes. In the learning process, all examples are extracted from the training data and grouped by taxonomical type (for example, duration, fully-specified date, underspecified date). Each group is processed separately, with the fully-specified expressions being first. The process starts by learning one-word expressions annotated in the `VAL` attribute as months. Then three-word expressions annotated at the level of day granularity are considered. In this way the meaning of some tokens can be learned. The output of the learner is a set of rules in a predefined format which together constitute a temporal tagger. Baldwin (2002) applied this mechanism to fully-specified temporal expressions, vague expressions referring to the past, present and future, weekday names, frequencies and durations. Underspecified expressions were not interpreted because of the missing date components. The learner did not support expressions of time granularity, temporal modifiers, half-years, quarter-years and year seasons, but the author claimed she could extend the learning mechanism to learn rules for such expressions too given more time for development. The number of rules learnt by the system was not specified.

While this work is a nice demonstration of the potential of such an example-based learner, it remains to be seen whether it could be used successfully to provide a tagger with the coverage of the whole TIMEX2 scheme. It is also not clear why we would need a learner to discover month or weekday names or that a four digit number denotes a year: encoding this knowledge by hand is not the most-time consuming part of the development of a rule-based system. The real drawback, however, is that there is still some knowledge engineering involved in designing the order of expression types which the algorithm uses for learning, the reasoning it performs over the training examples, and the ordering of the generated rules to be applied in the final tagger. The author notes that the results she obtained for English were lower than for French, because she worked primarily with French data during the development of the learning algorithm. This hints at the limits of the portability of the system across languages.

In building their tagger for Korean, Jang et al. (2004) used a dictionary method, which they refer to as rote learning. Given annotated training data, the system generates a dictionary with the strings that make up temporal expressions as the keys, and the encoding of their meaning before their interpretation in the context as values. Explicit temporal expressions always mean the same, so the collection of these is completely automatic; for context-dependent expressions, human intervention is necessary in order to encode the type, granularity and temporal offset: for example *last month* is encoded as `[M;REL;-1]`. When using the system on unseen data, the system would process only those expressions that had entries in the dictionary. This method was then augmented with a few simple manually-crafted rules to extend the tagger's coverage so that it would match, for example, the `NUMBER TEMPORAL_UNIT` pattern.

In order to reduce the amount of text a knowledge engineer has to read while carrying out empirical studies aimed at finding new vocabulary for extending a recognition grammar, Vichayakitti and Jaruskulchai (2005a) proposed the following seed-based human-assisted method.[33] First, a small set of seed temporal triggers is manually created (in the case of their tagger for Thai, the set contained 48 words such as the names of months, weekdays, parts of days, and names of temporal units). Then all phrases[34] in the development data (in their study 300 news articles were used) which contain any of these words are selected. The words from the phrases are checked against the dictionary of time-related vocabulary (which initially contains only the basic set of triggers); if a word does not appear in the dictionary it must be verified by a human and in case the engineer thinks it could be a part of a temporal expression it is added to the dictionary; if the word already appeared in the dictionary, it is omitted. The authors claim that by running this method on a set of 1000 news stories, this procedure reduced the size of text to be read by a developer by 85%; however, it is not reported how the resulting dictionary would differ in size and quality if it was created by a manual analysis of full texts.

2.6.3.2 Application of Machine-Learning Algorithms to Processing TEs

Machine learning was initially only applied to the task of temporal expression recognition; as Ahn et al. (2005b) note, all ML systems participating in ACE 2004 performed recognition only. This task is highly suitable for ML classifiers, as it can be seen as a supervised tagging problem: using for example the B-I-O model, for each token it has to be decided whether it is a (B)eginning, (I)nside or (O)utside of the expression; variations of this model can also be adopted. This approach was taken, for example, by Hacioglu et al. (2005) in their ATEL system and Ahn et al. (2007) in the TimexTag system.

We have not found any work that attempts to carry out the entire interpretation process using only machine-learning methods. Various hybrid solutions have been developed: Mani and Wilson (2000b), for example, suggested that a rule-based system could be extended with some machine-learned rules; they trained a C4.5 classifier to distinguish a vague use of *today* in the sense of *nowadays* (which did not get annotated in their framework) from the use meaning a specific date. Two rules learnt by the classifier were then integrated with the core, hand-developed grammar.

Wu et al. (2005a) identified 16 classes to which a temporal expression can belong. For each class they defined one interpretation rule (action). The classes are not mutually exclusive, so a temporal expression can belong to more than one class. For example, the expression *around 9 a.m. this Friday* belongs to three classes: class 2 (because of the *this Friday* component), class 11 (because of *around*), and class 12 (because of *9 a.m.*); see Table 2.17 for descriptions of these classes. The interpretation process now becomes a multi-label classification problem. For each class to which the expression is assigned, the associated action is triggered.

Another hybrid approach to the interpretation stage was designed by Ahn et al.

[33]We acknowledge this is not a machine-learning-based method; however, given its purpose of reducing the amount of work required to manually develop a recognition grammar, we present it

Class	Description	Interpretation Rule (Action)
2	Week-based format	Identify calendar date and week based expression for the reference date. Then infer the calendar date for this temporal expression and keep it as the value of attribute VAL.
11	Temporal expressions with modifiers	Retrieve the modifier and look up the dictionary. Keep the result as the value of attribute MOD.
12	Part of a day, month or year	Retrieve part of day/parts of month/part of year from this expression, look up the dictionary and keep the result as the value of attribute VAL.

Table 2.17: Example classes of temporal expressions and their corresponding interpretation rules used by Wu et al. (2005a).

(2007) and implemented in their TimexTag system. They applied ML-trained classifiers to two of the steps they identified in the interpretation stage. First, an expression is classified into one of five semantic classes: point, duration, generic/vague point, generic/vague duration and set. Then, knowing the type of the expression, a type-relevant grammar is applied to decompose the expression into constituents. In total there are 89 rules; the authors note that this is many fewer than in other rule-based systems, and that their rules are simpler. We have already noted that the comparison of recognition grammars in terms of rule counts is not very meaningful; however, our intuition is that a reduction in the size and complexity of the grammar is possible in this case because its functionality is considerably limited: the grammar does not have to recognize expressions in texts and capture their types. Their second classifier is used to determine the direction of interpretation of relative expressions. Obtaining the final value of the expression is carried out using relevant calendar arithmetics, leaving no space for a machine-learning approach here.

2.6.3.3 Typical Algorithms and Features

As the recognition of temporal expressions is turned into the problem of labelling and segmenting sequence data, algorithms which perform well for other such problems (for example, POS tagging, noun chunking or named entity recognition) are good candidates to use here. The most popular algorithm used for training the model is the Support Vector Machine (SVM), which was used, for example, by Hacioglu et al. (2005), Bethard and Martin (2006) and Ahn et al. (2007). Ahn et al. (2005a) explored Conditional Random Fields (CRF) and one of the ACE 2007 participants successfully used Maximum Entropy (MaxEnt) modelling. For the experiments taking the multi-label classification approach, Wu et al. (2005a) investigated two models: independent binary classification (IBC) and compared binary classification (CBC).

Features used by a classifier can be of different kinds: lexical, syntactic, semantic and external. Features considered in the literature (as found in (Hacioglu et al., 2005; Ahn et al., 2005a; Ahn et al., 2007) and descriptions of systems participating in the ACE 2007 evaluations) for the recognition task are:

lexical: the token itself, its lower-case version, its part-of-speech, use of a hyphen,

here.

[34]The paper does not specify what kinds of phrases were considered.

whether it is a number, frequency (discretized or not), suffix, prefix and lemma of the token and surrounding tokens, being listed in a gazetteer (for example, holiday names, month names, important dates);

syntactic: base phrase chunk, dependency heads, dependency relations between the token and its head;

semantic: semantic-role labels (from PropBank), WordNet information;

external: tags of temporal expressions identified by other taggers.

On its own, the lexical surface form is the most useful feature. Hacioglu et al. (2005) found that the external features, as used in the ATEL system, turned out to be the next most influential; however, these features require using another temporal expression tagger, which is not always possible. Excluding external features, POS tags are of next-highest importance for the performance of a classifier (which confirms the experience with rule-based systems, where rules also use POS tags).

The semantic type classification performed by Ahn et al. (2007) used the same features as for extent recognition. Determination of the direction of interpretation was modified by adding more problem-specific features: the tense of neighbouring verbs, and the comparison of day names, month names and years between the expression in question and the document time-stamp.

2.6.4 Evaluation and State-of-the-Art Performance

We will now look at the evaluation of temporal expression taggers, and discuss the results presented in the literature, and in particular the results from the ACE evaluations.

2.6.4.1 What Undergoes Evaluation?

Temporal expression taggers can be evaluated on different subtasks involved in the overall processing of temporal expressions. The standard areas on which results are reported are:

- **Detection**, which shows whether the existence of a temporal expression has been determined; in order to count as a detected expression, the system-generated annotation must have some extent overlap with an annotation in the gold standard. There are various ways one might define the required overlap; for example, in ACE 2004 one common character was enough, but in ACE 2005 and 2007 the overlap had to be at least 30% of the extents of both the system and reference annotations.

- **Bracketing**, or exact extent recognition: this measures in how many cases the extents of temporal expressions were identified correctly. An annotation is scored as correctly recognized only if its extent is identical to that of the answer key.

- **Normalisation**, or attribute–value assignment: this measures the number of values correctly assigned to the TIMEX2 attributes `VAL`, `MOD`, `SET`, `ANCHOR_VAL` and `ANCHOR_DIR`. Scores can be reported both separately for each attribute, or as an aggregate across all attributes.

2.6.4.2 Evaluation Metrics

The standard measures used for evaluation of the tagging systems are **precision** (P), **recall** (R), and **F-measure**; these are applied mainly to the recognition task, but can also take into account the values of the annotation tag attributes.

When evaluating the interpretation task, we can also use **accuracy** to measure the performance of a system. When TIMEX2 or a similar scheme is used, then accuracy is calculated for each attribute of those expressions which the system has correctly recognized (i.e. the system's output expression is matched with the gold-standard annotation). The value of the attribute can be either correct or incorrect. Accuracy then shows the proportion of the number of correct values to all generated, and **error rate** is the proportion of the number of incorrect values to all generated.

The ACE evaluations introduced the so-called **ACE value**, which combines evaluation of both recognition and interpretation and involves all attributes with a specified weighting of their importance. The maximum possible result is 100, and in the case of poor performance, for example, when the system generates many spurious annotations, the result can also be negative. The ACE value was defined differently for ACE 2004 than for ACE 2005 and ACE 2007, and in practice the 2005 and 2007 scoring tools provide slightly different results.

In Appendix C we provide precise definitions for all of these metrics.

2.6.4.3 The Performance of Existing TE Taggers

While most system descriptions found in the literature provide some form of evaluation, it is hard to directly compare the results of different systems. This is because the evaluation is often carried out in different set-ups, which includes using different evaluation metrics (for example, precision/recall/F-measure vs ACE value), processing different types of temporal expressions, and evaluating on different datasets which additionally can be of different genres. Different corpora can have different distributions of temporal expressions across types and therefore pose different levels of complexity in processing; for example, interpretation of context-dependent temporal expressions is much harder than processing fully-specified dates.

Most often the provided results concern extent recognition and filling the `VAL` attribute, but results for expression detection are also often included. From our observations, we conclude that F-measure for extent recognition is in the range of 55–90% and F-measure reported for the `VAL` attribute ranges from 50% to about 88%. But given the above differences in the evaluation set-ups, these numbers are of very limited utility for making direct comparisons. More useful are the results obtained in the ACE evaluations, which we present below. These evaluations used the same datasets and metrics for all participants, and the ACE corpora became standard datasets in the community, so even researchers not participating in ACE evaluations are able to present their results on the ACE training datasets (the test corpora are not publicly available).

An interesting result was shown by Ahn et al. (2007) for their SVM-based recognizer; although both precision and recall for detection were much lower than for their previous CRF-based system (Ahn et al., 2005a), both precision and recall were better for determining the extent of expressions; for details see rows E and H in Table 2.18.

As for the multi-classification approach developed by Wu et al. (2005a), although the IBC model performed better for classification (its F-measure was 93.18%, while for CBC it was 69.77%), the overall interpretation results were slightly in favour of the CBC model; for example, for the `VAL` attribute F-measure was 65.1% vs 64.0%. Although this work was originally carried out for temporal expressions in Chinese, the method can be used for other languages.

The ACE Results The official results for the ACE 2004 TERN task have not been published by the organizing committee. However, some participants have made available results which are said to be very close to those achieved at the official evaluations. Based on Hacioglu et al. (2005), Negri and Marseglia (2005), Ahn et al. (2005a), Verhagen et al. (2005) and Ahn et al. (2007), the unofficial results (for English) for the recognition stage are gathered in Table 2.18, and Table 2.19 contains results for interpretation. We can see that the top four scores for detection are very close to each other. A big difference can be observed between detection and bracketing for all systems, suggesting that finding the correct extent of an expression is a challenging task even if the expression has been detected. As for the interpretation results, Chronos, a rule-based system, obtained the highest results, especially for the `VAL` attribute (87.2% F-measure), which is the most important attribute in the scheme.

At ACE 2005 four participants submitted systems to the TERN task for English, and two for Chinese. The overall results, captured as the ACE Value score, for English ranged from 33.2 to 63.7, and the results for Chinese were 79.0 and 83.7; details across domains are shown in Table 2.20.[35] On average, the best results were obtained for newswire, the worst for broadcast conversations.

There were eight systems submitted for the TERN task at the ACE 2007 evaluations; four systems for English, four systems for Chinese and one for Spanish.[36] Two of the participants in the English evaluations (and one in the Chinese track) also participated in the ACE 2005 evaluations; one of the participants for English participated in both earlier ACE TERN tasks, which means they had a lot of experience with the task and the kind of data used in evaluations. The overall ACE values for English (six domains) were 48.3, 58.2, 59.4 and 61.6. The results across domains are shown in Table 2.21.[37] As at ACE 2005, the worst average performance was recorded on broadcast conversations. The best results were obtained for broadcast news, and newswire. The results both at ACE 2005 and 2007 seem to be lower than at ACE 2004, because the ACE value score penalizes systems for spurious matches (and for each attribute in a spurious expression there is an additional penalty calculated). For Chinese (newswire and weblog) the results were 1.0, 3.7, 14.8 and 40.4;[38] the difference between the best-performing system and the rest is large, because only this system carried out the interpretation of the expressions (the detection-only result for this system was 11.2). The system for Spanish (newswire only) achieved a score of 46.5.

[35]Source: `http://www.itl.nist.gov/iad/mig/tests/ace/ace05/doc/ace05eval_official_results_20060110.htm`.

[36]One participant submitted a system both for English and Chinese.

[37]Source: `http://www.itl.nist.gov/iad/mig/tests/ace/2007/doc/ace07_eval_official_results_20070402.html`.

[38]Because of some formal requirements which participants had to fulfil and which many failed, only the 14.8 result of one system was included in the official results.

2.6.4.4 Coverage of Grammars

All the taggers which we are aware of differ in the range of expressions they are capable of finding and interpreting. And it is not a matter of accuracy or flaws in their grammars or sequence labelling capabilities. Often some classes of temporal expressions are simply considered beyond the scope of the tagger, usually because of the application or domain for which the tagger was developed, but also for other arbitrary reasons, such as lack of time for further development. For example, Mani and Wilson (2000b) did not recognize durations (unanchored intervals) at all (this was later fixed when TempEx was extended to GUTime) and left generics uninterpreted; Schilder and Habel (2001), Schilder (2004), Saquete et al. (2003) and Jang et al. (2004) did not process set-denoting expressions; and Bittar (2009) did not correctly interpret (because of the lack of an appropriate treatment) weekday and month names and year parts.

Based on the error analyses presented in the literature, we find that the most common, and also most problematic, cases are context-dependent expressions. Context-dependent expressions require proper temporal focus tracking in order to provide the correct reference time. For example, *two days later* is interpreted by adding two days to the reference date. Bare weekday names, for example *Tuesday*, can be temporally placed either before or after the reference date, so determining the direction of interpretation is another complication. Generic expressions need to be distinguished from expressions referring to specific dates, but they often take the same surface lexical form and some contextual analysis is required. The extent of event-based expressions, such as *three days after the fire* or *the day he left*, are syntactically complex and therefore often a source of problems in recognition. Because of all these challenges, the typical class that the taggers are capable of recognizing and interpreting correctly are the fully-specified expressions, in particular those which refer to dates.

2.6.5 Summary

In this section we reviewed the literature concerning the construction of automatic taggers for temporal expressions. We found that there have been quite a few such systems reported in the literature, and two general techniques have been used in their development: the rule-based knowledge-engineering approach, and machine-learning using annotated data.

However, because there is only limited scope for machine learning to be applied in the interpretation stage, there is no system based entirely on this approach. It also turns out that in spite of there being reasonably large training corpora from a number of ACE evaluations, the vast majority of the systems are based entirely on manually-developed rules. At ACE 2004, all systems performing the full TERN task (i.e. including the interpretation stage) were completely rule-based; at ACE 2007, only one out of four participants used machine-learned classifiers in the interpretation stage, but even here this was still in combination with a number of interpretation rules.

The main objection raised against rule-based systems is that they are time-consuming to develop, and result in a monolithic structure. This may be true in many cases; in practice, however, there is no requirement to build rule-based systems as monoliths: some rule-based systems, for example those described by Negri and Marseglia (2005) and Vicente-Diez et al. (2008), have some degree of modularity. And, to be fair, the time required for development of training data necessary to train a statistical model should not be ignored.

		Detection			Bracketing		
Site	Sys Type	Precision	Recall	F-measure	Precision	Recall	F-measure
A	SVM	97.8	89.4	93.5	91.9	84.0	87.8
B	ML	97.3	89.1	93.0	90.5	82.9	86.5
C	rules	97.6	88.0	92.6	88.5	79.8	83.9
D	rules	97.6	88.0	92.6	88.5	79.8	83.9
E	CRF	97.9	85.6	91.4	85.5	74.8	79.8
F	rules	97.8	71.3	82.5	81.1	59.1	68.4
G	rules	98.7	61.7	75.9	84.3	52.7	64.8
H	SVM	92.9	81.3	86.7	87.8	76.9	81.9
I*	rules	–	–	85.0	–	–	78.0

Table 2.18: The scores for the recognition stage of the TERN task on ACE 2004 test dataset. Asterisk (*) denotes results obtained on the training dataset. Systems: A=ATEL, B=best performing recognition-only system at official ACE 2004 eval. (reporting after Ahn et al. (2007)), C=best performing full task system at official ACE 2004 eval. (reporting after Ahn et al. (2007)), D=Chronos, E=Ahn's CRF, F=Ahn's Rule Sys 1, G=Ahn's Rule Sys 2, H=TimexTag, I=GUTime.

	Chronos			Ahn's Rule Sys 2			Ahn's ML+Rules		
Attribute	P	R	F	P	R	F	P	R	F
VAL	87.5	87.0	87.2	84.5	44.9	58.6	79.3	50.1	61.4
ANCHOR_VAL	68.3	77.5	72.6	83.8	14.8	25.2	89.0	28.9	43.6
ANCHOR_DIR	83.3	69.8	76.0	85.1	15.0	25.5	90.4	29.4	44.4
MOD	83.7	72.0	77.4	—	—	—	—	—	—
SET	88.0	56.4	68.8	—	—	—	—	—	—

Table 2.19: The scores for the interpretation stage of the TERN task on the ACE 2004 test dataset.

Site	Overall	Broadcast Convers.	Broadcast News	Newswire	Telephone	UseNet Newsgroups	Weblogs
A	63.7	48.0	65.6	72.6	56.2	63.4	61.7
B	56.2	39.8	62.6	53.1	58.6	55.8	62.8
C	54.8	40.6	59.8	62.7	37.3	52.0	53.6
D	33.2	23.8	32.4	39.4	32.1	24.8	42.1

Table 2.20: The value scores for the TERN task at ACE 2005 (English).
A=Lang. Computer Corp., B=Lockheed Martin, C=Janya, D=Univ. of Amsterdam

Site	Overall	Broadcast Convers.	Broadcast News	Newswire	Telephone	UseNet Newsgroups	Weblogs
A	61.6	44.2	68.4	67.4	52.6	63.1	51.4
B	59.3	48.2	68.6	60.9	60.2	58.2	52.9
C	58.2	46.6	67.8	57.3	64.2	59.0	54.8
D	48.3	30.0	44.4	54.2	38.7	55.9	44.8

Table 2.21: The value scores for the TERN task at ACE 2007 (English).
A=Lockheed Martin, B=IBM, C=University of Amsterdam, D=Macquarie University

In the ACE 2004 evaluations, for the recognition task higher results were obtained by ML-based systems; this, however, does not necessarily tell us much about the superiority of one approach over the other one. In our view, this can be explained in the following way. The ACE 2004 TERN task was the first evaluation with such a wide range of temporal expressions to be extracted. The annotation guidelines occupy over 50 pages, so for participants using rule-based systems there was a high overhead effort required to learn what the task was about and to build an appropriate system. On the other hand, preparing a recognition system based on a machine-learning model required only retraining existing NER systems on a new dataset. Later, at ACE 2007, a rule-based system was already the best performing. Wu et al. (2005a) also reported a comparison of interpretation results between their two systems for Chinese: a rule-based system and one using multi-label classification of temporal expressions combined with one interpretation rule for each class. It turned out that, although the accuracy of the classification-based system was satisfactory, it was much lower than that of the rule-based system.

In favour of rule-based systems is the fact that, even if the expressions are recognised in text with a machine-learned tagger, the interpreter then needs to parse the expression anyway to build the semantics of its constituents. In a rule-based system this could already be carried out by the recogniser. Also it is not immediately clear what new features could be added to machine-learned taggers to significantly improve the performance, whereas rule-based systems can be always improved with new or more sophisticated rules.

We have not found any system that uses a hybrid recognizer, i.e. one where a module primarily based on machine-learning algorithms is augmented with some manual rules which handle special cases which are difficult to process correctly using a statistical approach (for example expressions which rarely occur in the training data); we also have not found a case where the module is primarily rule-based but is supported with some machine-learning classifiers which make some recognition decisions concerning the extent. Such a combination of the two approaches could be elaborated in more detail. The only exceptions we have seen are the use of temporal expressions found by a rule-based tagger as external features for an SVM-based recogniser (Bethard and Martin, 2006), and supporting the recogniser for Korean based on a learned dictionary with hand-written rules (Jang et al., 2004).

It should be noted that some systems use domain-specific simplifications. For example, Ahn et al. (2005a) limited the years to be recognized to those from the range 1900–2099. This limits spurious matches occurring in news articles, as it is uncommon for news to talk about events outside this period of time. Given that most taggers were developed primarily to process news articles, and that the evaluation is also very often carried out only on this domain, their performance for other data sources is by and large unknown. The ACE results revealed that there is a huge difference in taggers' performance between news and other domains, with news giving the best results. In their error analyses, authors usually list the most error-contributing types of expressions. Most often, these are context-dependent expressions, in particular weekday and month names, generic expressions, and event-based expressions.

Almost all systems which we are aware of generate TIMEX2 annotations, not TIMEX3, which is most likely because of the number of TIMEX2-annotated corpora and the ACE evaluation programs which had already run the TERN task. This confirms what we already suggested in Section 2.4, that TIMEX2 is the de facto standard for temporal expression annotation. Of course, there are also examples where the tagger by design does not generate any TIMEX-like annotation, but instead produces output in a format required by a specific application; for example, Treumuth (2006) generates SQL queries with temporal constraints.

2.7 Conclusions

In this chapter we have presented the current state of research related to the processing of temporal expressions, and provided some background material on the topic of temporal ontology. In Section 2.1 we noted that the term 'temporal expression' is used in the literature in two senses: a broader one, which includes any part of a sentence that conveys temporal information, and a narrower one, used in the information extraction community, which concerns most temporal adverbials and noun phrases. In our work we adopt the second sense.

Having informally set up the range of expressions we are interested in processing, in Section 2.2 we reviewed the types of temporal expressions that have been identified in the literature. Then, in Section 2.3, we looked at the semantic representations used by the taggers described in the literature.

In Section 2.4 we discussed the existing approaches to annotating temporal expressions, and in Section 2.5 we reviewed the existing available annotated corpora.

Finally, in Section 2.6 we reviewed the work concerned directly with the development of automatic tagging systems capable of recognizing and interpreting temporal expressions in texts. As in other areas of natural language processing, two dominant approaches can be observed: rule-based approaches and machine-learning approaches. We also reviewed the results achieved by these systems.

We take this review as a starting point for our own work presented in the remainder of this thesis. The main conclusions arising from the chapter are as follows:

- There are quite a few corpora with annotated temporal expressions, most of which were prepared for the ACE program, which we may use both for development and evaluation. Unfortunately, given the purpose of the ACE program, these corpora are somewhat limited: for example, they do not contain long narratives, in which temporal focus tracking would play a significant role in interpretation. For this reason, we prepare a new corpus, presented in Chapter 3, that consists of protracted narratives. The size of the corpus is comparable to the existing corpora, and the documents have temporal expressions annotated with TIMEX2 annotations. The choice of the scheme is based on its popularity in the research community that results in a number of resources that support the development and evaluation of our work.

- There is not a single taxonomy of temporal expressions widely adopted in the community. Although the solutions accepted by various authors are close to each other, there are differences and the terminology used varies. We also observed some deficiencies in each of the existing proposals. Therefore, we revise this area and propose in Chapter 4 a new comprehensive taxonomy.

- While both pattern-based and machine-learning based approaches have been used for the recognition of temporal expressions in text, a relatively unexplored area is the use of syntactic information in this task. In Chapter 5 we present two methods, dependency-based and constituency-based, that, given the trigger of a temporal expression as a seed, recognize the full extent of the expression by using a syntactic analysis of the sentence.

- There is not a particular semantic representation framework that appears distinctively better than other solutions, and the most common approach is to use a set of variables to first store context-independent information about the expression and then to derive the final representation of the referred-to entity. In Chapter 6 we propose a new representation for context-independent semantics of temporal expressions that can be used as an interface between recognition and interpretation modules of a system and is designed in a way that allows for convenient integration with the existing schemes for annotation of temporal expressions.
- A challenging problem in the interpretation of context-dependent temporal expressions is finding a reference time; additionally, many of these expressions require that the direction of the interpretation from the reference time is determined. These two problems have not been yet studied in-depth, and, in particular, various solutions reported in the literature have not been evaluated. Therefore, in Chapter 6 we experiment with various heuristics useful for solving these two problems and we provide comparative evaluations.
- Amongst the existing systems developed for the automatic processing of temporal expressions, the rule-based approach is dominant, with machine-learning being most useful for the recognition task. The results reported in the literature suggest that further improvements to the accuracy of the systems are possible. In Chapter 7 we present our rule-based system that, supported by an in-depth analysis of the problems appearing in the processing of temporal expressions, achieves the state-of-the-art accuracy for both the recognition and interpretation.

The high-quality broad-coverage automatic recognition and interpretation of temporal expressions in texts is a challenging task. One reason is the large number of forms temporal expressions can take. Secondly, even more problems arise in the interpretation stage, in which the meaning of temporal expressions needs to be processed in the context of the rest of the document. In the remaining part of the thesis we present our work that contributes to the development of methods and resources useful in the automatic processing of temporal expressions.

Chapter 3

The WikiWars Corpus

Our investigation of existing corpora for research on the interpretation of temporal expressions revealed that the bulk of available data is skewed with regard to the types of text that are represented. Although, overall, we find a variety of text types are represented (transcripts of broadcast conversations, telephone conversations, face-to-face conversations about scheduling meetings, transcripts of broadcast news, newswire, UseNet discussions, and weblog entries), we noted that there are very few extended documents describing events that unfold over protracted periods. This has the consequence that most context-dependent temporal expressions can be successfully interpreted using the document time-stamp as the reference time. This simplification of the problem prohibits the exploration of cases where temporal expressions have to be interpreted with respect to a reference time found elsewhere in the body of the text; in such cases the selection of a reference time requires some form of a temporal focus tracking mechanism. Essentially, the existing corpora with annotated temporal expressions do not support sufficiently the development of such mechanisms.

We also found that the corpora being used as benchmark data contain a significant number of annotation errors. These problems skew the evaluation of temporal taggers, make error analysis much harder, and leave doubts about the reliability of any statistics derived about the types of temporal expressions present in the data based on the gold-standard annotations.

Given these concerns, we decided to develop a high-quality corpus that was richer in the kinds of temporal expressions that have so far been relatively ignored in this area of research. By doing this, we also wanted to obtain some hands-on experience with the annotation task to see how suitable the TIMEX2 scheme is for the annotation of temporal expressions in other kinds of texts. For the content of the corpus, we chose those sections of selected Wikipedia articles that present descriptions of the course of wars and conflicts. This chapter describes the construction of this corpus and discusses a variety of issues that arose in its development.

3.1 The Limitations of Existing Corpora

In Section 2.5 we reviewed those existing corpora that are annotated with temporal expressions. These were two MUC corpora, the TIDES corpus, several ACE corpora and the TimeBank corpus.

The disadvantage of the MUC corpora is that they are annotated with TIMEX, which is significantly different from the newer TIMEX2 and TimeML annotation schemes. This is a result of the NER task definition used in the MUC evaluations, where temporal expressions were required to be detected, but not interpreted. The range of expressions annotated in the MUC corpora are also somewhat limited (for example, many context-dependent temporal expressions were excluded from the task) compared to the requirements of many NLP tasks, and to TIMEX2 and TimeML.[1] Therefore, we do not find the MUC corpora suitable for our current work.

The TIDES corpus is annotated with the first version of TIMEX2 (Ferro et al., 2001) so its use in our research is also not ideal, and the TimeBank corpus is annotated with TimeML, which in consequence stops us from using this corpus for evaluation of our TIMEX2-based algorithms. However, they are still valuable resources which we may use to investigate the use of temporal expressions in real texts; this is so because, unlike the MUC datasets, these two corpora have a wider range of temporal expressions annotated.

The ACE evaluations have become something of a benchmark in the information extraction community for many tasks, including the TERN task. The relevant corpora are often used as standard resources for estimating the performance of temporal expression taggers. From the range of available ACE corpora, we will primarily use the ACE 2005 Training and ACE 2007 Evaluation datasets. The ACE 2004 Development corpus is annotated with a slightly different version of the TIMEX2 scheme than the corpora from ACE 2005 and 2007, so its usability is limited. The ACE 2007 Evaluation corpus is based on the ACE 2005 Evaluation corpus, thus it makes sense to evaluate a tagger on the bigger dataset.

However, both the ACE 2005 Training and ACE 2007 Evaluation datasets have limitations. In particular, the documents in these corpora tend to be limited in length and, in consequence, discourse structure. This impacts on the number, range and variety of temporal expressions they contain. Existing research carried out on the interpretation of temporal expressions, e.g. by (Baldwin, 2002; Ahn et al., 2005b; Mazur and Dale, 2008), suggests that many temporal expressions in documents, especially news stories, can be interpreted fairly simply as being relative to a reference time that is the document time-stamp. This phenomenon does not carry over to longer, more narrative-style documents that describe extended sequences of events, as found, for example, in biographies or descriptions of protracted geo-political events.

3.2 Creating WikiWars

In this section we report on the process of creating the WikiWars corpus, which was developed to overcome some of the limitations just noted. We start with a description of how we selected the content for the corpus documents, how we extracted and

[1]We discussed the range of the annotated temporal expressions in TIMEX, TIMEX2 and TimeML in Section 2.4.

preprocessed the text from the source Wikipedia articles, and how we prepared the gold-standard annotations. We also present our observations concerning some issues with using TIMEX2 as an annotation scheme for temporal expressions, and, finally, we provide various statistics on the size of the WikiWars corpus as compared to other existing corpora.

3.2.1 Selecting Data Sources

To avoid copyright issues that might arise in the development and distribution of any corpus we might develop, we decided to use Wikipedia as a source. After considering various types of historical narrative, we settled on descriptions of the course of wars and conflicts as being particularly rich in the kinds of phenomena we wanted to explore (e.g. multiple events being presented, multithreaded discourse structure, and the need to find the reference time within the narrative). The corpus, which we call WikiWars, consists of 22 documents sourced from English Wikipedia; each document describes the historical course of a war. Despite the small number of documents, their length means that the corpus yields a large number of temporal expressions—2681 in total, making it larger than many of the corpora currently used in the community—and poses new challenges for tracking temporal focus through extended texts. The corpus has been made available for others to use.[2]

The selection of articles was based on the following procedure.

1. We first submitted to Google the phrase query 'most famous wars in history' and we chose the top-ranked result; this linked to a web page at the Yahoo Answers site where web users proposed their lists of the ten most famous wars in history.[3] Site-registered visitors to the page provided votes on the proposals; we used the list that had received the most votes. This listed the names of ten wars considered to be the most important in the whole of history.

2. Then we submitted to Google another phrase query for 'the biggest wars'. We again chose the top-ranked result, which linked to a web page listing the names of the 20 biggest wars that occurred in the 20th century, measured in terms of the number of military deaths they involved.[4]

3. We combined the two lists, eliminated duplicates (four wars), and searched Wikipedia for articles describing these wars.[5] Wikipedia did not contain an article for one war (the Riffian War);[6] and we considered two articles (about the First Sudanese Civil War and the Chaco War) as inappropriate for our purposes since they did not describe the course of these wars, but only provided some general

[2]See `http://www.TimexPortal.info/WikiWars`.

[3]The page is located at `http://answers.yahoo.com/question/index?qid=20090209222618AAauMN1` (last accessed on 2009-12-20).

[4]The page is located at `http://users.erols.com/mwhite28/war-list.htm` (last accessed on 2009-12-20).

[5]In some cases the two web pages that listed the wars referred to the conflicts with alternative names to those used in the titles of the Wikipedia articles; for example, the *Persian Wars* is a common means of referring to the *Greco-Persian Wars* (499–450 BC) and the *Biafran War* refers to the same conflict as the *Nigerian Civil War* (1967-1970).

[6]Much later, after we had finished preparing the corpus, we discovered that Wikipedia did in fact have an article about this war, but under the name 'The Rif War'. However, in any case the article was too short to be useful, and did not present the course of the war.

Name of the war	Year span	The document part of the URL
World War II	1939–1945	`World_War_II`
World War I	1914–1918	`World_War_I`
American Civil War	1861–1865	`American_Civil_War`
American Revolutionary War	1775–1783	`American_Revolutionary_War`
Vietnam War	1955–1975	`Vietnam_War`
Korean War	1950–1953	`Korean_War`
Iraq War	2003–...	`Iraq_War`
French Revolution	1789–1799	`French_Revolution`
Persian Wars	499–450 BC	`Greco-Persian_Wars`
Punic Wars	264–146 BC	`Punic_Wars`
Chinese Civil War	1945–1949	`Chinese_Civil_War`
Iran-Iraq War	1980–1988	`IranIraq_War`
Russian Civil War	1917–1923	`Russian_Civil_War`
French Indochina War	1946–1954	`First_Indochina_War`
Mexican Revolution	1911–1920	`Mexican_Revolution`
Spanish Civil War	1936–1939	`Spanish_Civil_War`
French-Algerian War	1954–1962	`Algerian_War`
Soviet-Afghanistan War	1979–1989	`Soviet_war_in_Afghanistan`
Russo-Japanese War	1904–1905	`Russo-Japanese_War`
Russo-Polish War	1919–1921	`PolishSoviet_War`
Biafran War	1967–1970	`Nigerian_Civil_War`
Abyssinian War	1935–1936	`Second_Italo-Abyssinian_War`

Table 3.1: The source of the articles used in WikiWars (the full URLs can be resolved by preceding each document part of the address with `http://en.wikipedia.org/wiki/`).

information about the nature of the conflicts. Also, the second chosen web page listed two periods of the Chinese Civil War separately, but we found a single article about all the periods of this war.[7]

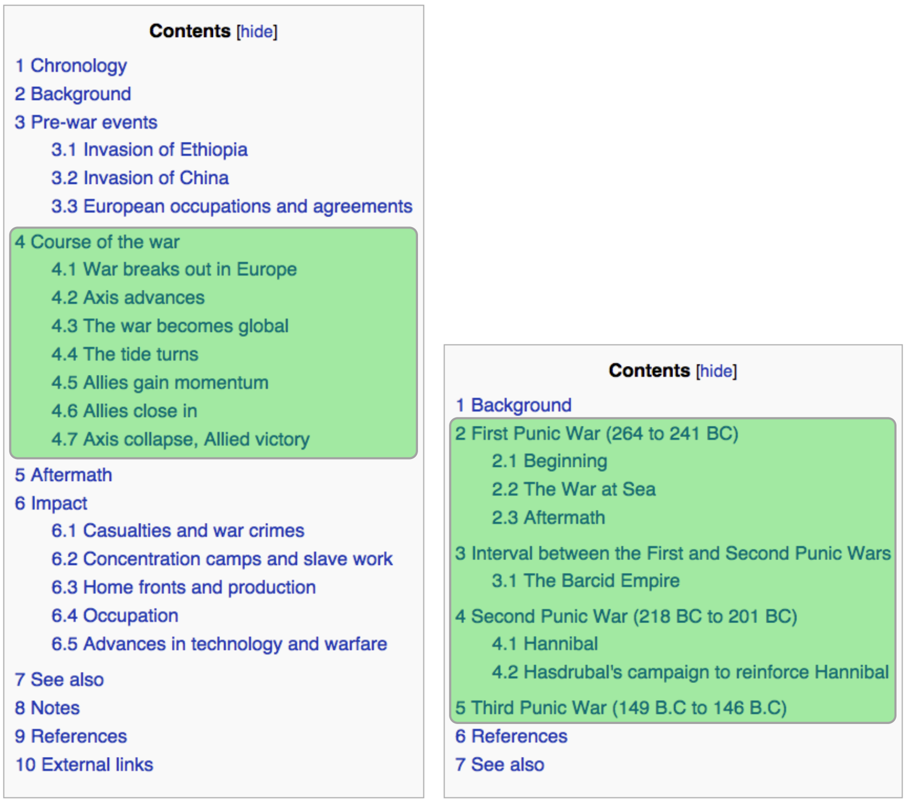
This process resulted in 22 articles; the URLs to the relevant Wikipedia pages are provided in Table 3.1, where apart from the name of the war we also give the time span involved, to enable clear identification of the conflict.

3.2.2 Text Extraction and Preprocessing

Each of the 22 selected Wikipedia articles provided content for one document in the corpus. However, we only used those sections of the articles that described the historical course of the wars; depending on the length of the article and the content of the sections, this meant leaving out smaller or larger parts of the texts.

In Figure 3.1 we present the tables of contents of two of the articles (*World War II* and the *Punic Wars*), and we use shaded areas to mark those sections which were used to provide the content for the corpus. As we can see, some articles (e.g. *World War II*) have a section titled *Course of the war* which contains exactly the type of text we are interested in. The preceding sections, although they may also contain historical

[7]Formally, the conflict lasted from 1927 to 1991, but there were periods of no fighting or fighting together against a common external enemy; consequently, the civil war is divided into periods.



Contents [hide]

1 Chronology
2 Background
3 Pre-war events
 3.1 Invasion of Ethiopia
 3.2 Invasion of China
 3.3 European occupations and agreements
4 Course of the war
 4.1 War breaks out in Europe
 4.2 Axis advances
 4.3 The war becomes global
 4.4 The tide turns
 4.5 Allies gain momentum
 4.6 Allies close in
 4.7 Axis collapse, Allied victory
5 Aftermath
6 Impact
 6.1 Casualties and war crimes
 6.2 Concentration camps and slave work
 6.3 Home fronts and production
 6.4 Occupation
 6.5 Advances in technology and warfare
7 See also
8 Notes
9 References
10 External links

Contents [hide]

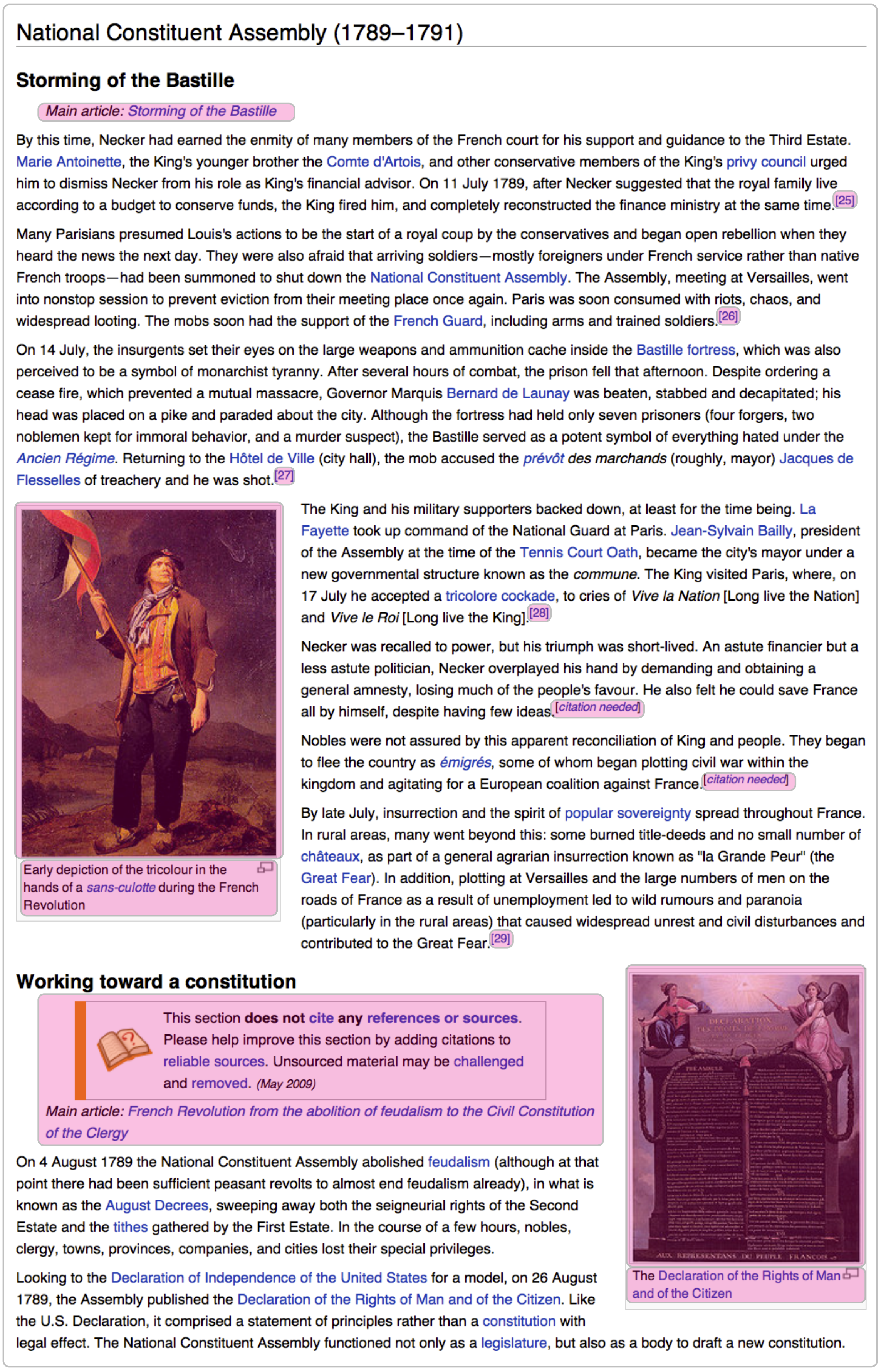
1 Background
2 First Punic War (264 to 241 BC)
 2.1 Beginning
 2.2 The War at Sea
 2.3 Aftermath
3 Interval between the First and Second Punic Wars
 3.1 The Barcid Empire
4 Second Punic War (218 BC to 201 BC)
 4.1 Hannibal
 4.2 Hasdrubal's campaign to reinforce Hannibal
5 Third Punic War (149 B.C to 146 B.C)
6 References
7 See also

Figure 3.1: The table of contents of the Wikipedia articles about WW2 (left) and the Punic wars (right); the shaded areas show which sections were included in the WikiWars documents.

narratives and temporal expressions, are usually quite short and do not make a coherent story: they only highlight some selected events or episodes that happened before the war (often these may be the direct and indirect reasons for war breaking out). The sections following the description of the course of the war, such as *Aftermath* or *Impact*, do not present a course of events, but rather discuss the statistics of the conflict, and provide a summary and conclusions concerning the political situation in the region after the conflict. Sections like *See also* or *References* are of course outside our interest as well.

In some other cases (e.g. the *Punic Wars*) the article may be relatively short and not contain a dedicated section presenting the course of the war. This may be so, for example, when the conflict (e.g. the Punic wars, the civil war in China) consists of a number of clearly separated periods of peace and war time which are described in separate sections of the article. In such situations we chose more than one section to include in the corpus version of the document. A rule that we adhered to in all cases was that we did not use only parts of sections; i.e. each section was either included in a WikiWars document as a whole, or was completely rejected.

To prepare the corpus, we first manually copied text from the selected sections and pasted them into text files. This involved the manual removal of picture captions and cross-page links with anchor text like *Further information* or *For more details on this topic*. We then ran a Perl script over the results of this extraction process to



National Constituent Assembly (1789–1791)

Storming of the Bastille

Main article: Storming of the Bastille

By this time, Necker had earned the enmity of many members of the French court for his support and guidance to the Third Estate. Marie Antoinette, the King's younger brother the Comte d'Artois, and other conservative members of the King's privy council urged him to dismiss Necker from his role as King's financial advisor. On 11 July 1789, after Necker suggested that the royal family live according to a budget to conserve funds, the King fired him, and completely reconstructed the finance ministry at the same time.[25]

Many Parisians presumed Louis's actions to be the start of a royal coup by the conservatives and began open rebellion when they heard the news the next day. They were also afraid that arriving soldiers—mostly foreigners under French service rather than native French troops—had been summoned to shut down the National Constituent Assembly. The Assembly, meeting at Versailles, went into nonstop session to prevent eviction from their meeting place once again. Paris was soon consumed with riots, chaos, and widespread looting. The mobs soon had the support of the French Guard, including arms and trained soldiers.[26]

On 14 July, the insurgents set their eyes on the large weapons and ammunition cache inside the Bastille fortress, which was also perceived to be a symbol of monarchist tyranny. After several hours of combat, the prison fell that afternoon. Despite ordering a cease fire, which prevented a mutual massacre, Governor Marquis Bernard de Launay was beaten, stabbed and decapitated; his head was placed on a pike and paraded about the city. Although the fortress had held only seven prisoners (four forgers, two noblemen kept for immoral behavior, and a murder suspect), the Bastille served as a potent symbol of everything hated under the *Ancien Régime*. Returning to the Hôtel de Ville (city hall), the mob accused the *prévôt des marchands* (roughly, mayor) Jacques de Flesselles of treachery and he was shot.[27]

Early depiction of the tricolour in the hands of a *sans-culotte* during the French Revolution

The King and his military supporters backed down, at least for the time being. La Fayette took up command of the National Guard at Paris. Jean-Sylvain Bailly, president of the Assembly at the time of the Tennis Court Oath, became the city's mayor under a new governmental structure known as the *commune*. The King visited Paris, where, on 17 July he accepted a tricolore cockade, to cries of *Vive la Nation* [Long live the Nation] and *Vive le Roi* [Long live the King].[28]

Necker was recalled to power, but his triumph was short-lived. An astute financier but a less astute politician, Necker overplayed his hand by demanding and obtaining a general amnesty, losing much of the people's favour. He also felt he could save France all by himself, despite having few ideas.[*citation needed*]

Nobles were not assured by this apparent reconciliation of King and people. They began to flee the country as *émigrés*, some of whom began plotting civil war within the kingdom and agitating for a European coalition against France.[*citation needed*]

By late July, insurrection and the spirit of popular sovereignty spread throughout France. In rural areas, many went beyond this: some burned title-deeds and no small number of châteaux, as part of a general agrarian insurrection known as "la Grande Peur" (the Great Fear). In addition, plotting at Versailles and the large numbers of men on the roads of France as a result of unemployment led to wild rumours and paranoia (particularly in the rural areas) that caused widespread unrest and civil disturbances and contributed to the Great Fear.[29]

Working toward a constitution

This section **does not cite any references or sources.** Please help improve this section by adding citations to reliable sources. Unsourced material may be challenged and removed. *(May 2009)*

Main article: French Revolution from the abolition of feudalism to the Civil Constitution of the Clergy

On 4 August 1789 the National Constituent Assembly abolished feudalism (although at that point there had been sufficient peasant revolts to almost end feudalism already), in what is known as the August Decrees, sweeping away both the seigneurial rights of the Second Estate and the tithes gathered by the First Estate. In the course of a few hours, nobles, clergy, towns, provinces, companies, and cities lost their special privileges.

The Declaration of the Rights of Man and of the Citizen

Looking to the Declaration of Independence of the United States for a model, on 26 August 1789, the Assembly published the Declaration of the Rights of Man and of the Citizen. Like the U.S. Declaration, it comprised a statement of principles rather than a constitution with legal effect. The National Constituent Assembly functioned not only as a legislature, but also as a body to draft a new constitution.

Figure 3.2: A fragment of the Wikipedia article about the French Revolution; the shaded areas mark the parts of the text that were not included in the corresponding WikiWars document.

convert some Unicode characters, such as ligatures, spaces, apostrophes, hyphens and other punctuation marks, into ASCII. This was followed by running another script to remove a variety of elements that were not relevant for our purposes:

- citation references of the form of `[n]`, `[n]:x`, and `[n]:x-y`, where `n` is a reference number and `x, y` are page numbers within the referred-to publication; and
- a range of other Wikipedia annotations such as, for example, *[citation needed]*, *[neutrality disputed]* or *[clarification needed]*.

In Figure 3.2 we present a fragment of the article about the French Revolution with shaded areas showing those parts that were removed in the manual and automatic cleaning of the text.

Finally, we converted each of the text files into an SGML file; each document was wrapped in one `DOC` tag, inside which there are `DOCID`, `DOCTYPE` and `DATETIME` tags. The document time-stamp is the date and time at which we downloaded the page from Wikipedia to our local repository.[8] The proper content of the article is wrapped in a `TEXT` tag; Figure 3.3 illustrates the structure, which intentionally follows that of the ACE 2005 and 2007 documents, so as to make the processing and evaluation of the WikiWars data highly compatible with the tools used to process the ACE corpora.[9]

3.2.3 Creating Gold-Standard Annotations

Manually annotating any kinds of named entities in a document collection from scratch is a time-consuming, laborious and error-prone process. Having prepared our input SGML documents, we decided to first process them with our DANTE temporal expression tagger (which we present in detail in Chapter 7).

DANTE outputs the original SGML documents augmented with an inline TIMEX2 annotation for each temporal expression found. These output files can be imported to Callisto,[10] an annotation tool developed by Day et al. (2004) that supports TIMEX2 annotations. Using a temporal expression tagger as a first-pass annotation tool not only significantly reduces the amount of human annotation effort required (creating a tag from scratch requires a number of clicks in the annotation tool), but also helps to minimize the number of errors that arise from overlooking markable expressions through ‘annotator blindness’.

[8]The goal of having a document time-stamp is to provide a reference time for interpretation of deictic temporal expressions (i.e. those whose interpretation is relative to the time of writing). However, as conventional historical narratives do not typically contain deictics, the `DATETIME` tag should have no influence for temporal processing of the WikiWars texts. We also acknowledge that instead of taking as the time-stamp the time of downloading the article, we could also use the time-stamp of the last edit made to the article; we note, however, that in many cases these two time-stamps are the same, because many of the Wikipedia articles are modified as often as every day, so again the chosen strategy would practically make no difference. Moreover, given the way these articles are edited, what we really needed to correctly interpret deictics would not be the time-stamp of the last change made in the article, but for each temporal expression in question we would need to store the time-stamp of the last modification of the sentence encapsulating the expression. Unfortunately, the `DATETIME` tag is designed to store one time-stamp only for the document as a whole.

[9]Note that there are slight differences between documents in different domains of the ACE corpora.

[10]See `http://callisto.mitre.org`.

```
<DOC>
<DOCID>08_FrenchRev</DOCID>
<DOCTYPE SOURCE="wikipedia"> HISTORY ARTICLE </DOCTYPE>
<DATETIME> <TIMEX2 val="2009-12-19T17:00:00">2009-12-19T17:00:00</TIMEX2>
</DATETIME>
<TEXT>
[...]
The King and his military supporters backed down, at least for <TIMEX2
val="PRESENT_REF" anchor_val="1789-07-14" anchor_dir="AS_OF">the time being</TIMEX2>.
La Fayette took up command of the National Guard at Paris.  Jean-Sylvain Bailly,
president of the Assembly at <TIMEX2 val="1789-06-20">the time of the Tennis Court
Oath</TIMEX2>, became the city's mayor under a new governmental structure known
as the commune.  The King visited Paris, where, on <TIMEX2 val="1789-07-17">17
July</TIMEX2> he accepted a tricolore cockade, to cries of Vive la Nation [Long live
the Nation] and Vive le Roi [Long live the King].

Necker was recalled to power, but his triumph was short-lived.  An astute financier
but a less astute politician, Necker overplayed his hand by demanding and obtaining
a general amnesty, losing much of the people's favour.  He also felt he could save
France all by himself, despite having few ideas.

Nobles were not assured by this apparent reconciliation of King and people.  They
began to flee the country as migrs, some of whom began plotting civil war within the
kingdom and agitating for a European coalition against France.

By <TIMEX2 val="1789-07" mod="END">late July</TIMEX2>, insurrection and the spirit
of popular sovereignty spread throughout France.  In rural areas, many went beyond
this:  some burned title-deeds and no small number of chteaux, as part of a general
agrarian insurrection known as "la Grande Peur" (the Great Fear).  In addition,
plotting at Versailles and the large numbers of men on the roads of France as a
result of unemployment led to wild rumours and paranoia (particularly in the rural
areas) that caused widespread unrest and civil disturbances and contributed to the
Great Fear.

Working toward a constitution

On <TIMEX2 val="1789-08-04">4 August 1789</TIMEX2> the National Constituent
Assembly abolished feudalism (although at that point there had been sufficient
peasant revolts to almost end feudalism already), in what is known as the <TIMEX2
val="1789-08">August</TIMEX2> Decrees, sweeping away both the seigneurial rights
of the Second Estate and the tithes gathered by the First Estate.  In the course of
<TIMEX2 val="PTXH" anchor_val="1789-08-04" anchor_dir="WITHIN">a few hours</TIMEX2>,
nobles, clergy, towns, provinces, companies, and cities lost their special
privileges.

Looking to the Declaration of Independence of the United States for a model, on
<TIMEX2 val="1789-08-26">26 August 1789</TIMEX2>, the Assembly published the
Declaration of the Rights of Man and of the Citizen.  Like the U.S. Declaration,
it comprised a statement of principles rather than a constitution with legal effect.
The National Constituent Assembly functioned not only as a legislature, but also as
a body to draft a new constitution.
[...]
</TEXT>
</DOC>
```

Figure 3.3: The structure of a document in the WikiWars corpus with a fragment of text corresponding to the Wikipedia article presented in Figure 3.2.

The annotations produced by DANTE were then manually corrected in Callisto via the following process.[11]

1. Annotator 1 (the author of this thesis) corrected all the annotations produced by DANTE, both in terms of extent and the values provided for TIMEX2 attributes. Crucially, this step also included the annotation of any temporal expressions missed by the automatic tagger: a careful reading of documents was facilitated by computer-aided search for names of temporal units (e.g. *month*, *weekend*, and other time-related tokens). Spurious matches were also removed at this stage.

2. Then, Annotator 2 (the supervisor of the author of the thesis) read each of the marked-up texts to check the revised annotations, with a particular objective of identifying any temporal expressions that remained unidentified by the above process. This resulted in a list of errors found and a number of doubts or queries in regard to potentially problematic annotations.

3. Annotator 1 then verified and fixed the errors, after discussion in those cases where there was a disagreement.

The final SGML files containing inline annotations were then transformed into ACE APF XML annotation files, this being the stand-off markup format developed for ACE evaluations. This transformation was carried out using the `tern2apf` tool developed by NIST for the ACE 2004 evaluations, with some modifications introduced by the author of this thesis to adjust the tool to support ACE 2005 documents and to add a document ID as part of the ID of a TIMEX2 annotation; this was done so that all annotations would have corpus-wide unique IDs.

The resulting corpus is thus available in two formats: one contains the original documents enriched with inline annotations (as shown in Figure 3.3), and the other consists of stand-off annotations in the ACE APF format (see Figure 3.4).

3.2.4 Observed Deficiencies of TIMEX2

The annotation process described above revealed some issues with the use of TIMEX2 in practice. First, the flexibility of the TIMEX2 scheme, which can at first be seen as an advantage, actually makes it ambiguous in use. One instance of this phenomenon relates to the fact that the TIMEX2 guidelines state that the provision of some attribute values for event-based expressions (such as *three weeks after the siege of Boston began* or *the first year of the American invasion*) is optional. Since our corpus has a significant number of such expressions, the decision as to whether or not to provide semantic values in such cases has a potentially large impact on the perceived performance of a tagger. In such cases, we decided only to provide the value when it is very clear from the article itself what the value should be. In this way, the corpus provides for

[11] We recognize that our procedure is a bit different from the methodology often used in corpus linguistics, where first we have two annotators working independently over the whole corpus and then a third person (a referee) combines the results into final gold-standard annotations. Unfortunately, we did not have access to the resources required to follow this approach here. Therefore, instead of having a third person acting as a referee, which is the most important difference between the two procedures, we resolved any differences between the two annotators via extensive discussions and argumentation based on careful analysis of the annotation guidelines. We believe that this variation in the methodology did not have any negative impact on the quality of the created resource.

```
<?xml version="1.0"?>
<!DOCTYPE source_file SYSTEM "apf.v5.1.2.dtd">
<source_file URI="08_FrenchRev.key.xml" SOURCE="wikipedia" TYPE="text">
<document DOCID="08_FrenchRev">

[...]

<timex2 ID="08_FrenchRev-T36"  ANCHOR_DIR="AS_OF" ANCHOR_VAL="1789-07-14" VAL="PRESENT_REF">
  <timex2_mention ID="08_FrenchRev-T36-1">
    <extent>
      <charseq START="8575" END="8588">the time being</charseq>
    </extent>
  </timex2_mention>
</timex2>
<timex2 ID="08_FrenchRev-T37" COMMENT="EVENT-BASED" VAL="1789-06-20">
  <timex2_mention ID="08_FrenchRev-T37-1">
    <extent>
      <charseq START="8700" END="8732">the time of the Tennis Court Oath</charseq>
    </extent>
  </timex2_mention>
</timex2>
<timex2 ID="08_FrenchRev-T38"  VAL="1789-07-17">
  <timex2_mention ID="08_FrenchRev-T38-1">
    <extent>
      <charseq START="8850" END="8856">17 July</charseq>
    </extent>
  </timex2_mention>
</timex2>
<timex2 ID="08_FrenchRev-T39"  MOD="END" VAL="1789-07">
  <timex2_mention ID="08_FrenchRev-T39-1">
    <extent>
      <charseq START="9513" END="9521">late July</charseq>
    </extent>
  </timex2_mention>
</timex2>
<timex2 ID="08_FrenchRev-T40"  VAL="1789-08-04">
  <timex2_mention ID="08_FrenchRev-T40-1">
    <extent>
      <charseq START="10091" END="10103">4 August 1789</charseq>
    </extent>
  </timex2_mention>
</timex2>
<timex2 ID="08_FrenchRev-T41"  VAL="1789-08">
  <timex2_mention ID="08_FrenchRev-T41-1">
    <extent>
      <charseq START="10283" END="10288">August</charseq>
    </extent>
  </timex2_mention>
</timex2>

[...]

</document>
</source_file>
```

Figure 3.4: An APF stand-off annotation file with selected annotations corresponding to the Wikipedia article presented in Figure 3.2.

the evaluation of a more sophisticated tagger which handles text-derived information about events as well as information about temporal expressions.

Another area where TIMEX2 is not ideal is in regard to the annotation of time zones. First, only whole-hour time differences are supported, which eliminates some time zones (e.g. Afghanistan lies in UTC+04:30). Second, according to the TIMEX2 guidelines, time-zone information is supposed to be marked only for expressions which contain an explicit statement of the time zone. However, we can often infer from the context that subsequent unadorned time references should inherit the same time zone as an earlier time reference. Nonetheless, we followed the TIMEX2 guidelines in this regard; fortunately, WikiWars does not contain many expressions that incorporate time-zone information.

We also found that, in a not insignificant number of cases, it is impossible to provide a precise and correct value for a temporal expression. For example, the TIMEX2 guidelines stipulate that the anchor of a duration cannot have a `MOD` attribute. This means that, if the anchor is the expression *mid-August*, the value of the anchor must refer only to August; consequently, the semantics of *mid-* is lost.

TIMEX2 only supports nonspecific expressions which have explicit information about granularity. Expressions such as *a very short time* or *a short period of time* therefore cannot be provided with any value, since the context does not indicate whether the period involved should be measured in days, weeks, or months. One might consider using the typical durations of events of the corresponding types in such cases, but this solution also presents problems; see (Pan et al., 2006) for a discussion.

As is acknowledged in the TIMEX2 guidelines, the treatment of set expressions (i.e. recurring times and durations and frequencies, such as *twice a month*) is underdeveloped. One rule states that set expressions should not be anchored (Ferro et al., 2005, p. 42); this has the consequence that the full semantics of the expression *annually since 1955* cannot be provided, and the expression is therefore treated as two separate expressions, *annually* and *1955*.

TIMEX2 distinguishes between the temporal and non-temporal uses of *now*. There are 74 instances of this term in WikiWars; however, in many cases it is hard to decide whether the expression is being used temporally or not. We decided to annotate them all, but did not provide a value for those expressions in those cases where it was not clear what the value should be.

Finally, alternative calendars are not supported by TIMEX2, so an expression like *February in the pre-revolutionary Russian calendar* cannot receive a value unless it appears in an appositive construction which provides an alternative description. Similarly, consider Example (3.1):

(3.1) On *9 November 1799* (18 Brumaire of *the Year VIII*) Napoleon Bonaparte staged the coup of 18 Brumaire which installed the Consulate.

Here, *18 Brumaire of the Year VIII* is a date in an alternative calendar used in 18th-century France, but we annotated only *the Year VIII* based on the trigger *year*. Note that *18 Brumaire* also occurs later in the sentence, but is not annotated.

Corpus	# Docs	Size (KB)	# Tokens	# TEs	# Tokens / # TEs	# TEs / # Doc
ACE 2005 Training	599	1,733	318,785	5,469	58.3	9.13
TIDES	95	257	42,932	3,541	12.1	37.27
WikiWars	22	633	119,468	2,681	44.6	121.86
ACE 2007 Evaluation	254	567	104,779	2,028	51.7	7.98
TimeBank v1.2	183	570	78,444	1,414	55.5	7.73
ACE 2005 Evaluation	155	356	63,217	1,154	54.8	7.45

Table 3.2: Statistics for the WikiWar corpus compared to those for other corpora; TEs = Temporal Expressions.

3.2.5 Corpus Statistics

The corpus contains 22 documents with a total of almost 120,000 tokens and 2,681 temporal expressions annotated in the TIMEX2 format.[12] In Table 3.2 we compare the WikiWars corpus with the other available corpora of texts marked-up with temporal expressions. While the ACE 2005 Training corpus remains the largest corpus, WikiWars is larger than each of the ACE 2005 and 2007 evaluation corpora and the TimeBank v1.2 corpus, both in terms of number of tokens and annotated temporal expressions. WikiWars has an order of magnitude more temporal expressions in each document, and a slightly higher density of temporal expressions overall than the other corpora except the TIDES corpus, in which on average there is one temporal expression every 12.1 tokens (this high density is a consequence of the domain of the corpus, which is conversations about scheduling a meeting; the majority of sentences concern the date or time of a meeting).

Table 3.3 presents statistics on the individual documents that make up the corpus. The documents vary considerably in size, the smallest (16_SpanishCivilWar) consisting of only 1,455 tokens, and the largest (05_VietnamWar) being eight times larger at 11,640 tokens. The density of TIMEX2 annotations varies from 1 in 23.1 to 1 in 72.1 tokens (see the grey-backgrounded cells in Table 3.3), but for the majority of documents this ratio lies between 30 and 60.

3.3 The Nature of Wikipedia Articles

Wikipedia articles may be edited by a large number of people over a significant number of revisions. We checked how often the articles that make up the WikiWars corpus were modified in the period from January 2008 to February 2010. On average, each article was changed almost 52 times per month, with the monthly number of changes for a single article ranging from 1 to 372.[13] The minimum average number of edits per

[12]All token counts presented in Tables 3.2 and 3.3 were obtained using GATE's default English tokeniser; hyphenated words, e.g. *British-held* or *co-operation*, were counted as single tokens.

[13]Note that these numbers are for the articles as a whole, and not just the sections which we extracted (although these are usually the major part of the article). Additionally, these edits include both major changes (e.g. adding a new section), minor changes (e.g. correcting a grammar error or adding a comma), vandalism (deletion of the page content or the on-purpose provision of false information) and restoring the page after an act of vandalism has been detected.

Document ID	# Tokens	# TIMEX2	# Tokens / # TIMEX2
01_WW2	5,593	170	32.9
02_WW1	10,370	265	39.1
03_AmCivWar	3,529	75	47.1
04_AmRevWar	5,695	147	38.7
05_VietnamWar	11,640	245	47.5
06_KoreanWar	5,992	149	40.2
07_IraqWar	8,404	247	34.0
08_FrenchRev	9,631	175	55.0
09_GrecoPersian	7,393	129	57.3
10_PunicWars	3,475	57	61.0
11_ChineseCivWar	3,905	102	38.3
12_IranIraq	4,508	98	46.0
13_RussianCivWar	3,924	104	37.7
14_FirstIndochinaWar	3,085	71	43.5
15_MexicanRev	3,910	78	50.1
16_SpanishCivilWar	1,455	63	23.1
17_AlgerianWar	7,716	130	59.4
18_SovietsInAfghanistan	5,306	110	48.2
19_RussoJap	2,760	62	44.5
20_PolishSoviet	5,137	106	48.5
21_NigerianCivilWar	2,091	29	72.1
22_SecondItaloAbyssinianWar	3,949	69	57.2
Total	119,468	2,681	44.6
Average per document	5,430	122	–
Standard deviation	2,663	64	–

Table 3.3: Statistics for the WikiWars corpus.

month for an individual document was 13.08 (17_AlgerianWar), and the maximum was 171.77 (07_IraqWar). Table 3.4 presents a summary of the number of changes made to individual articles and the corpus as a whole.

The nature of the revision process in Wikipedia leads to some artefacts that may not be typical of other document sources, such as news, where the text is usually carefully prepared by its author and checked by an editor. This is not to say that Wikipedia content is necessarily of a lower quality; this is an encyclopedia with many people and bots controlling its quality, and there exist manuals of style for authors to help them avoid errors and ambiguity and to ensure maximum consistency.[14] However, given the large number of editors with various degrees of fluency and experience in writing and editing, it would not be surprising if some parts of the texts are not perfect. In the process of preparing the gold-standard annotations for the WikiWars corpus, we have observed the following issues.

[14]The main manual of style for Wikipedia is located at `http://en.wikipedia.org/wiki/Wikipedia:Manual_of_Style`; with respect to our specific interests, the specific guidelines concerning formatting dates and numbers can be found at `http://en.wikipedia.org/wiki/Wikipedia:DATE`.

Document	Min	Median	Average	Max
01_WW2	33	88.50	105.42	239
02_WW1	23	53.50	66.19	201
03_AmCivWar	7	34.50	43.04	99
04_AmRevWar	9	40.00	50.38	133
05_VietnamWar	16	48.00	74.73	148
06_KoreanWar	30	94.50	101.54	338
07_IraqWar	25	170.00	171.77	372
08_FrenchRev	2	14.00	29.19	250
09_GrecoPersian	10	34.50	39.31	94
10_PunicWars	5	41.00	37.31	72
11_ChineseCivWar	6	31.00	31.04	68
12_IranIraq	33	57.00	72.35	163
13_RussianCivWar	7	16.00	18.12	29
14_FirstIndochinaWar	2	16.50	16.92	62
15_MexicanRev	2	29.50	32.92	76
16_SpanishCivilWar	21	57.00	62.85	146
17_AlgerianWar	2	12.00	13.08	28
18_SovietsInAfghanistan	27	72.00	76.69	169
19_RussoJap	14	24.50	29.50	76
20_PolishSoviet	1	10.00	13.58	36
21_NigerianCivilWar	4	13.50	13.42	30
22_SecondItaloAbyssinianWar	4	18.50	43.00	340
Corpus	1	35.00	51.92	372

Table 3.4: Statistics concerning the number of changes made per month to the Wikipedia articles in the WikiWars corpus, for the months from January 2008 to February 2010.

3.3.1 Broken Narratives

In some articles we found situations where a sentence does not appear to cohere with those on either side of it. This may be the result of a number of modifications being made by different authors, or it may be due to a lack of writing skill on the part of the person who wrote the paragraph in question. Example (3.2) provides an example of this phenomenon.

(3.2) ALN commandos committed numerous acts of sabotage in France in *August*$_{[1958]}$, and the FLN mounted a desperate campaign of terror in Algeria to intimidate Muslims into boycotting the referendum. Despite threats of reprisal, however, 80 percent of the Muslim electorate turned out to vote in *September*$_{[1958]}$, and of these 96 percent approved the constitution. In *February 1959*, de Gaulle was elected president of the new Fifth Republic. He visited Constantine in *October*$_{[1958]}$ to announce a program to end the war and create an Algeria closely linked to France.

The sentence about de Gaulle being elected president contains a temporal expression (*February 1959*) which progresses the temporal focus in the narrative to 1959, but the later context of the article strongly suggests that the subsequent reference to *Octo-*

ber in the following sentence is in fact October 1958.[15] To get the correct semantic value, a tagger would have to interpret this expression with the reference to the expression (*September*) preceding what appears to be an interpolated sentence, a feat of interpretation that is easily missed by human readers.

3.3.2 Ambiguous Writing

We have also found instances of lack of precision in writing which lead to ambiguous statements. Consider Example (3.3), sourced from the document 18_SovietsInAfghanistan:

(3.3) The Afghan government, having secured a treaty in *December 1978* that allowed them to call on Soviet forces, repeatedly requested the introduction of troops in Afghanistan in *the spring* and *summer of 1979*. They requested Soviet troops to provide security and to assist in the fight against the mujahideen rebels. On *April 14, 1979*, the Afghan government requested that the USSR send 15 to 20 helicopters with their crews to Afghanistan, and on *June 16*, the Soviet government responded and sent a detachment of tanks, BMPs, and crews to guard the government in Kabul and to secure the Bagram and Shindand airfields. In response to this request, an airborne battalion, commanded by Lieutenant Colonel A. Lomakin, arrived at the Bagram Air Base on *July 7*. They arrived without their combat gear, disguised as technical specialists. They were the personal bodyguards for President Taraki. The paratroopers were directly subordinate to the senior Soviet military advisor and did not interfere in Afghan politics.

After *a month*, the Afghan requests were no longer for individual crews and subunits, but for regiments and larger units. In *July*, the Afghan government requested that two motorized rifle divisions be sent to Afghanistan. *The following day*, they requested an airborne division in addition to the earlier requests.

Here, in the first paragraph there are four temporal expressions related to the Afghan government asking for troops and equipment. There is also one date (*June 16*) related to the Soviet reply to these requests and the sending of tanks, and one date (*July 7*) related to the arrival of an airborne battalion. The second paragraph starts with the expression *after a month*; the first interpretation we might consider is that this is a month after the 7th July mentioned in the previous paragraph, i.e. the month would end on or around the 6th of August. But the following sentence reveals that this is not the case, as it mentions some requests for larger units that were also made in July. Usually a narrative progresses forwards in time, not backwards, so it is much more likely that the elapsed month referred to started around either 14th April or 16th June. If the second sentence elaborates the first one, then it is a month from 16th June; if it just mentions one of the requests for larger units, then it is probably a month from 14th April.

It is also unclear whether the second paragraph talks about the same request for airborne forces which was mentioned in the first paragraph: both these events are dated July. The phrase *In response to this request* is in fact placed very oddly, as its preceding sentence does not mention any request, but rather talks about the Soviet response to requests. This may suggest that what at first looks just like a careless and ambiguous use of the expression *after a month* is in fact indicative of a larger problem of lack of coherency in these two paragraphs.

[15]We have added subscripts in the example to indicate the actual year being referred to.

3.3.3 Restarting the Time of Narrative

Another property that WikiWars exhibits, and which is noticeably absent from the simpler ACE data, is what we might think of as a discourse mechanism for resetting the temporal focus in a text. This is a feature of complex texts in general, rather than something that is specific to Wikipedia as a source. In these cases, the discourse does not follow a single global timeline from the beginning to the end of the document, but is rather divided into subdiscourses which describe separate chains of events that often have a common temporal starting point. This is typical in the description of big, often international, conflicts, where one can distinguish several theaters of the war, e.g. the eastern and western theaters, where events happen in parallel.

In most cases the switch to a different 'part of the story' can be determined not only by analysing the events and their geographic locations, but by recognizing that the first date appearing in the new subdiscourse is generally fully specified. This is, however, not always the case, as shown in the following example extracted from the article 01_WW2:

(3.4) In *September 1944*, Soviet Red Army troops advanced into Yugoslavia and forced the rapid withdrawal of the German Army Groups E and F in Greece, Albania and Yugoslavia to rescue them from being cut off. By this point, Communist-led partisans under Marshal Josip Broz Tito controlled much of the territory of Yugoslavia and were engaged in delaying efforts against the German forces further south. In northern Serbia, the Red Army, with limited support from Bulgarian forces, assisted the partisans in a joint liberation of the capital city of Belgrade on *October 20*$_{[1944]}$. *A few days later*, the Soviets launched a massive assault against German-occupied Hungary that lasted until the fall of Budapest in *February 1945*. In contrast with impressive Soviet victories in the Balkans, the bitter Finnish resistance to the Soviet offensive in the Karelian Isthmus denied the Soviets occupation of Finland and led to the signing of Soviet-Finnish armistice on relatively mild conditions with subsequent Finland's shift to the Allied side.

By *the start of July*$_{[1944]}$, Commonwealth forces in Southeast Asia had repelled the Japanese sieges in Assam, pushing the Japanese back to the Chindwin River while the Chinese captured Myitkyina. In China, the Japanese were having greater successes, having finally captured Changsha in *mid-June*$_{[1944]}$ and the city of Hengyang by *early August*$_{[1944]}$. Soon after, they further invaded the province of Guangxi, winning major engagements against Chinese forces at Guilin and Liuzhou by *the end of November*$_{[1944]}$ and successfully linking up their forces in China and Indochina by *the middle of December*$_{[1944]}$.

In the first paragraph of the example, the text discusses military actions concerning the Red Army in the Balkans from September 1944 to February 1945; it also discusses the situation in Finland but no new time expressions are given, so we might expect the events concerning Finland occurred in the same time frame as the events in the Balkans. The second paragraph presents the fighting in Asia that involved the Commonwealth, China and Japan; these events have no direct link to those described in the first paragraph. However, the paragraph starts with the underspecified expression *the start of July*, in which information about the year is missing. If we assumed that the paragraph continues the story started in the first paragraph, we would get an incorrect interpretation of this expression as referring to 1945. Instead, the reader must realise that this temporal expression starts a new subdiscourse. Clearly, quite sophisticated processing is required to handle this phenomenon adequately. An additional challenge is the fact that the temporal expressions at the beginning of the two paragraphs refer to different months, so it is not even the case that when restarting the time of the global narrative (to July 1944) we go back to the previously saved reference time (September 1944); we need to go back in time beyond that point. We also note that the subsequent temporal expression, *mid-June*, goes even further back in time, as it also concerns 1944.

3.3.4 The Use of Deictic Expressions

One of the articles in the corpus, 07_IRAQWAR, contained a number of deictic expressions, indicative of the fact that the events described were happening contemporaneously to the time of writing (as is often the case in news stories). Here are two examples:

(3.5) a. Democrats plan to push legislation *this spring* that would force the Iraqi government to spend its own surplus to rebuild.

b. A protester said that despite the approval of the Interim Security pact, the Iraqi people would break it in a referendum *next year.*

Obviously, after some time these expressions will no longer make sense, or will be open to misinterpretation by an unwary reader. Unlike, for example, news stories, there is no 'at-the-time-of-writing' time-stamp associated with these sentences: for the reader of a Wikipedia article, the only reference date is the time of reading. As it turns out, the two sentences above were written in April and December 2008, respectively; but few readers would be likely to consult the document's edit history to determine that this is the case.[16] Arguably, these sentences should be corrected, making the temporal expressions fully-specified (e.g. *in spring of 2009* and *in 2009*), or context-dependent (e.g. *in spring of that year* and *the following year*) if there is a context in the article which supports their correct interpretation. Of course, not only the temporal expressions need to be revised, but also the tense and aspect of the verbs used in the sentences. In the gold-standard annotations, however, we provided the values by interpreting these expressions with respect to the document time-stamp (i.e. `2010-SP` and `2010`), as the text itself does not provide any evidence that other dates were intended, and to provide interpretations that require more subtle reasoning would arguably place unfair expectations on any automated interpretation process.

3.3.5 The Use of Time-Zone Information

We noticed that time-zone information is used very rarely in WikiWars. This may be because it is usually assumed that any times provided in an article are local times, i.e. they are in the time zone of the place described in text. However, it then remains unclear how we should interpret times when there are mentions of places from different time zones; this is not uncommon.

A particularly confusing situation occurs in the 05_VIETNAMWAR document, in which there are two temporal expressions at the granularity of minutes. The first one is provided without any time zone (*3:00 a.m* in Example (3.6a)), but the second one, introduced in a different context a few sections later, indicates *local time* (*11:30 a.m. local time* in Example (3.6b)).

[16]It takes somewhat laborious document archeology to extract this information from Wikipedia's archive.

(3.6) a. Escalation of the Vietnam War officially started on *the morning of 31 January 1965*, when orders were cut and issued to mobilize the 18th Tactical Fighter Wing from Okinawa to Da Nang Air Force Base (AFB). A red alert alarm to scramble was sounded at Kadena AFB at *3:00 a.m.*

b. On *30 April 1975*, VPA troops overcame all resistance, quickly capturing key buildings and installations. A tank crashed through the gates of the Presidential Palace, and at *11:30 a.m. local time* the NLF flag was raised above it.

This introduces a confusion about the first expression, since it is unclear whether or not it is in the same time zone as the second expression. Another issue that an automated tagger needs to face here is that the second expression does not explicitly point to a specific time zone: it is up to the tagger to figure out what geographical area is meant by 'local', and then in what time zone this area is located. The requirement to determine a time zone based on geographical clues may be important not only for Wikipedia articles, but also for many other genres, including news.

The second example we draw attention to here is Example (3.7), which comes from article 01_WW2.

(3.7) On *December 7 (December 8 in Asian time zones), 1941*, Japan attacked British and American holdings with near simultaneous offensives against Southeast Asia and the Central Pacific.

The italicized temporal expression is difficult to detect, and it is not clear how it should be annotated. But it is also imprecise with respect to which time zone is intended: Asia encompasses ten time zones. Therefore it is impossible to fully interpret the expression. Note also that the expression combines a time zone with a date, rather than with a time. While uncommon, this is not incorrect; but the TIMEX2 guidelines do not explicitly allow for this circumstance.

3.3.6 Quotes Missing a Time-Stamp

Occasionally it happens that an article contains a quoted utterance, but there is no indication of when the utterance was made. For example, in the document 05_VIETNAM-WAR we find the following:

(3.8) Nixon said in an announcement, "I am *tonight* announcing plans for the withdrawal of an additional 150,000 American troops to be completed during *the spring of next year*. This will bring a total reduction of 265,500 men in our armed forces in Vietnam below the level that existed when we took office *15 months ago*."

It is impossible to determine what dates are meant by the three temporal expressions present in the announcement. In some cases this information may be provided in citation footnotes, but this is not always the case; when this is absent, such expressions can only be annotated at the level of textual extent and a localised, context-dependent semantics.

3.3.7 Grammatical Errors

Given that Wikipedia articles can be modified by practically anyone with internet access, one might expect that the articles would contain more grammatical errors than typical news text. In fact, we found relatively few grammatical errors; however, some of these do unfortunately impact on the interpretation of temporal expressions. For example, consider the following sentence:

(3.9) This was the city's fourth conquest in *a years' time*, leaving it a ruin; the 1.5 million pre-war population was down to 200,000, and the people were suffering from severe food shortages.

Here, the expression *a years' time* is incorrectly punctuated; it should be *a year's time*. There is a deeper problem, however; the expression itself is actually misused. Given the context of the article, it seems that the sentence should be corrected to 'This was the city's fourth conquest within *a year*'.

3.4 Conclusions

In this chapter we have introduced a new corpus, which we call WikiWars, containing temporal expressions annotated using the most recent version of TIMEX2. Although the corpus consists of only 22 documents, it contains 2,681 temporal expressions; this is more than are found in many popular benchmark corpora used by the research community.

The novelty of the corpus, and the major motivation for its creation, lie in the domain and genre of the documents. As they describe the temporal progression of military conflicts, they are of a considerable length and have a high density of temporal expressions. The documents also exhibit a more interesting discourse structure than is typically found in the currently available corpora. This makes WikiWars a particularly valuable resource for studying the problem of tracking the temporal focus through a discourse, which is a required ability if we are to locate the reference times needed to interpret context-dependent expressions.

We have made the corpus publicly available in the hope that it would be useful for the community as a whole. Indeed, Strötgen and Gertz (2011b) have already created a parallel version of the corpus for German, which opens new research possibilities for the construction of multilingual temporal expression taggers.

Chapter 4

A Taxonomy of Temporal Expressions

Before we attempt to design and implement any algorithm for the processing of temporal expressions, we need to start with an understanding of the variety of temporal expressions that appear in real texts. In this chapter we characterise a taxonomy of the various different kinds of temporal expressions that are to be found in texts, and describe them in detail. In doing so, we not only analyse the ways temporal expressions may refer to temporal entities and establish the terminology we will use throughout the thesis, but we also make some important distinctions that will play a key role in our approach to the interpretation of temporal expressions, based on the observation that expressions of different types require different approaches. A major source of insights underlying the creation of this taxonomy was our experience with corpus annotation gained in the creation of the WikiWars corpus described in the previous chapter.

Our focus here is on temporal expressions in English text; however, we believe that most of the underlying semantic observations should also hold for many other languages. We start by introducing in Section 4.1 some basic concepts, which we will use not only when presenting the taxonomy but also in the rest of the thesis.

In Section 4.2 we define what we accept as a temporal expression and we present the main types we distinguish in our taxonomy. Then, in Sections 4.3–4.5 we go on to discuss in detail all the subtypes that make up the taxonomy, and in Section 4.6 we draw attention to some semantic distinctions concerning temporal expressions of various taxonomical types.

4.1 Basic Concepts

We define and discuss three basic concepts to which we will refer throughout this thesis; these are the notions of a **temporal unit**, **granularity** and a **timeline**.

When referring to time, we use temporal units for identification and quantification: the units provide a means for locating events in time and expressing how long things take. The size of temporal units can be conceptualised as a collection of granularities, over which there is a partial ordering. A timeline is a common representation for representing chronology, i.e. arranging events in their order of occurrence in time. The level of detail shown on a timeline depends on the underlying temporal unit that has been chosen. We will also use the notion of a timeline to distinguish some types of temporal expressions; in particular, some expressions refer to temporal entities that cannot be marked on a timeline.

Temporal Units Quantifying time allows us to compare the durations of processes, to express the amount of time that passes between events, and to determine when things occur. In order to quantify time, we make use of culturally long-established named units, such as the notion of a year. The status of these units has been formalised in the International System of Units (SI),[1] which defines a 'second' as the 'base unit of time'; other units, referred to as 'derived', are expressed as multiples or submultiples of the base SI-unit, so that, for example, one minute is 60 seconds, and one day is 86400 seconds.

The traditional source of definitions for many temporal units are various astrophysical phenomena. For example, a 'year' is popularly defined as 'a period of one revolution of the earth around the sun'. Its average duration is 365.256363004 SI days; a year defined in this way is called a 'sidereal year'. But there are also other notions of a year, such as the 'tropical (solar) year', the 'anomalistic year' and the 'lunar year', each having a different definition and average duration. These notions are of great significance for astronomy, but in everyday life we use the conventional calendar as a system for organizing days for social, religious, commercial, and administrative purposes.

The current de facto world-wide standard is the Gregorian calendar, which uses the 'Julian year'; this is defined as having an average value of exactly 365.25 SI days. Because the average length involves a fraction, for practical purposes the calendar introduces special rules for expressing the duration of one unit in terms of an integral number of finer-grained units.[2] For example, in the Gregorian calendar a year is either 365 or 366 days and, as none of these two values produces an integer value when divided by 12, a month can consist of 28, 29, 30 or 31 days. It is really up to the calendar to determine how to divide up time and what units to introduce; for example, in the the Judeo-Christian tradition a week is a period of seven days, but in other cultures a week consists of as little as three days to as many as ten days.[3]

[1]SI, which is an international abbreviation having its origins in the French name *Système International d'Unités*, standardises a number of units of measurement, for example the meter, kilogram, and second. See `http://www.bipm.org/en/si`.

[2]Obviously, we would not want to celebrate the new year at midnight one year, and then at six in the morning the next year, and at noon the following year.

[3]In ancient Egypt the calendar was based on a ten-day week; a year consisted of twelve three-week-long months, leaving five extra days at the end of each 360-day-long year.

Calendars also assign identifiers to specific periods of time that arise when dividing one unit of time into smaller units. The identifier may be a number (e.g. the year 2000), a name (e.g. *January* for the first month of any given year), or a complex structure (e.g. the identifier given to each day is a 'date', which consists of the number of a year, a month within that year, and a day within that month).

In the work described here we will use the common definitions of the temporal units associated with the Gregorian calendar, the conventional business calendar, the system of four annual seasons, and other popular units used in contemporary talk about time. Of course, for some particular domains or applications, other temporal units could be taken into account as well.

Types of Temporal Units Temporal units are of a number of different kinds. Some units have clear boundaries and constant durations, and can be expressed by means of other units using a single consistent ratio. For example, a day is always equivalent to 24 hours, which in turn is always 1440 minutes or 86400 seconds; as an element of a calendar system, a day always starts and ends at midnight, which is a clearly determined time. Similarly, a century always begins in the year $nn01$ and ends in the year $mm00$, where $mm = nn + 1$. In Figure 4.1, which diagrammatically relates all the temporal units considered in our work, these well-defined units appear in solid-edged frames and are connected to other well-defined units with solid arrows; the labels on the arrows indicate the number of units of one type that make one unit of the second type. These ratios are constant.

Other units also have clear boundaries, but have varying durations. This is true, for example, of the notion of a month, which can have 28, 29, 30 or 31 days; a quarter-year lasts 90, 91 or 92 days; a half-year is 181, 182 or 184 days; a year can have 365 or 366 days; and a decade lasts 3652 or 3653 days. Although a given temporal unit may be of varying duration with respect to a second temporal unit, as in the cases just described, it may consist of constant number of other units; for example, although the number of days in a year varies, it always consists of 12 months. In Figure 4.1, we indicate with dashed-line frames those units which are of varying duration with respect to at least one other unit, and we mark the relations of varying duration with dashed-line arrows. Although a month as a unit has varying duration, a specific month in a calendar has well-defined boundaries. For example, the first month of each year always has 31 days; it always starts on the first day of the year, and it lasts until the 31st day of the year. February and any other month is also unambiguously located in the calendar system, because although it can have either 28 or 29 days, we have a well-defined algorithm for checking whether or not a given year is a leap year. Similarly, although the first half of a year can last either 181 or 182 days, the boundaries are well-defined and always the same: the two halves of the year always begin on 1st January and 30th June.

Finally, there are also temporal units which have both **vague** durations and vague temporal location. Examples of these are the annual seasons, the parts of a day, and the notion of the weekend; in Figure 4.1 they are presented in frames bounded by a dotted line. Of course one could provide precise definitions of these units: the astronomical definition of winter states that it begins on the solstice and ends on the equinox, which means that the northern hemisphere winter starts on either (depending on the year) 21 or 22 December and ends on 20 or 21 March. However, in everyday natural language communication, references to seasons are rarely taken to refer to precisely these dates: the more conventional understanding is that winter is simply a period

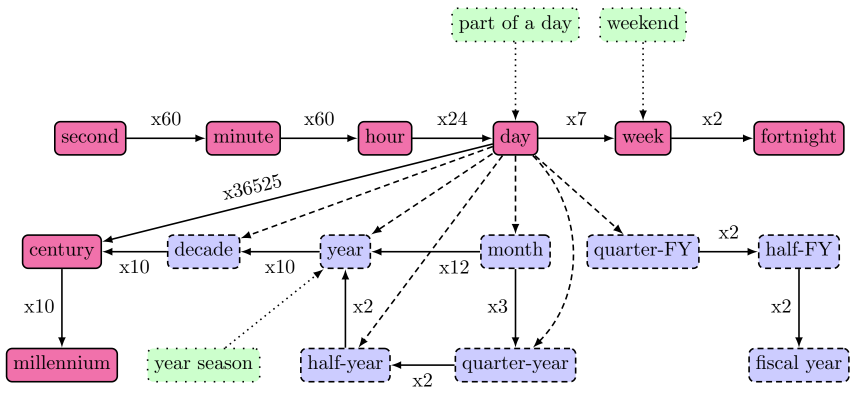


Figure 4.1: Temporal units and the relationships between them.

from around the beginning of December to around the middle of March. Similarly, for some applications one might define specific parts of a day quite precisely: we might stipulate that 'morning' is from 5am to noon, 'afternoon' is from noon to 6pm, 'evening' from 6pm to 10pm, and 'night' from 10pm to 5am the following day. However, such precise definitions are not in widespread use, and we consider it not unreasonable for different people to have different views on where the borderlines lie. 'Weekend' is a similarly vague temporal unit.

Granularity When an utterance mentions, for example, an event, and gives its temporal location, it may do so more or less precisely. For example, an individual can indicate when she was born in many different ways, as shown in Example (4.1).

(4.1) a. I was born in *1982*.

b. I was born in *June 1982*.

c. I was born on *20th June 1982*.

d. I was born at *2:13pm on 20th June 1982*.

Obviously, using a different temporal expression does not impact the actual time at which the event being referred to took place; it is just that the time of the event is approximated by the temporal value of the temporal expression used.

The change in the level of precision can be considered as a change in the **granularity** of the expression. In this thesis we will occasionally compare two expressions in terms of their granularity; this notion will play an important role in the design of our interpretation algorithms.

We can treat granularity as a function which returns the size of the granule associated with a given temporal expression.[4] We will denote the values returned by the function as G indexed by an identifier that corresponds to a particular temporal unit, e.g. G_{hour} is the granularity of an hour and G_{week} is the granularity of a week. There is a partial ordering over these values. When comparing two expressions we will refer

[4]We note that the notion of granularity has received significant attention in the field of artificial intelligence; however, exploration of the formal representation of granularity is beyond the scope of the present work. The foundations of the formalisation of granularity were laid by Hobbs (1985); Euzenat (1995) developed an algebraic approach to represent granularity conversions, and Bettini et al. (2000) describes the application of granularity systems to databases and temporal reasoning.

to the more specific time as being of **finer** granularity and to the less-specific time as being of **coarser** granularity. For the temporal units considered in our work we then have the following ordering:

(4.2) $G_{sec} < G_{min} < G_{hour} < G_{daypart} < G_{day} < G_{weekend} < G_{week} < G_{fortnight} < G_{month} < G_{quarteryear} < G_{halfyear} < G_{year} < G_{decade} < G_{century} < G_{millennium}$

Note that the above ordering also includes two vague temporal units, daypart and weekend. The granularity of a year season lies between the granularity of a month and a half-year, but as we cannot precisely compare the granularities of a quarter-year with a year season, we need to order seasons separately:

(4.3) $G_{month} < G_{yearseason} < G_{halfyear}$

The granularities can also be equal. Financial years, for example, are of the same granularity as calendar years: $G_{FY} = G_{year}$.

In our work, we will adopt the view, as taken in TIMEX2, that the granularity of a temporal expression is equal to the granularity of the finest temporal unit used in the expression.[5] Therefore, the granularity of the expression in Example (4.1a) is G_{year}, and the granularity in Example (4.1d) is G_{min}.

Timelines Temporal expressions refer to temporal entities, which can be represented by temporal values: these may either describe the temporal location of the entity in time, or the duration of the entity, which can be seen as the temporal distance between two temporal locations. These temporal locations are drawn from the domain of time, T, which we can think of as a **universal timeline**. The points on such a timeline are the ontological instants. Treating time as a continuum, such a timeline is isomorphic to the real numbers.

When we refer to time in natural language, however, we use elements of a calendar, which are temporal units; consequently, we always refer to time at some granularity. In the following example the year granularity G_{year} is used:

(4.4) The first version of the standard was published in *1997*.

Temporal units are far from being durationless instants. For convenience, it is useful to consider having a number of **point-based timelines** with the points being at the granularity of specific temporal units; for example, T_{year} is a timeline whose points are at the granularity of a year (G_{year}). Timelines can be easily visualised as number lines; in Figure 4.2 we present an example of a T_{year} timeline. Note that each of the events is attached to one of the points on the timeline, and no events can lie between the points; unlike the universal timeline, the T_{unit} timelines are isomorphic to integer number lines. In the rest of the thesis we will refer to point-based timelines simply as timelines.

4.2 What Counts as a Temporal Expression?

For the purpose of discussing temporal expressions, we distinguish two types of temporal entities that these expressions may refer to: points on a timeline, and periods. Correspondingly, we distinguish two kinds of temporal expressions:

[5]We acknowledge that such an approach will later have negative consequences for interpretation of some temporal expressions. If we consider the sentence *I bought this car over 2 years ago*, and assume it was uttered in October 2010, it needs to be interpreted as 'before 2008', where what it really says is 'before October 2008'. Unfortunately, for the purposes of evaluation we are committed to the choices made in TIMEX2.

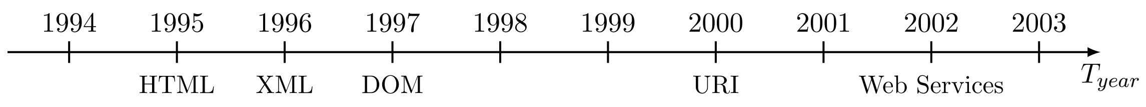


Figure 4.2: An example timeline of a year granularity (based on `http://www.w3.org/2005/01/timelines/timeline.pdf`).

1. **Point-referring expressions**: these are references which indicate *when* something happened or will happen. Here are some examples:

   (4.5) a. Thomas Edison invented the electric light bulb in *1879.*
   b. John was born on *July 1, 1967.*
   c. The incident occurred on *Friday.*
   d. He will go to Norway *next week.*
   e. *Last century* brought us some of the most horrible wars.
   f. I remember I saw him *the day I bought a new car.*

   Although we call these expressions point-referring expressions (or **points** for short), it should be borne in mind that we mean a point on a specific type of timeline T_{unit}, and not the abstract point of zero duration (an ontological instant). Therefore there is nothing wrong in considering, in appropriate circumstances, a century to be a point, although intuitively we think of a century as a period of time stretching over one hundred years.

   In some cases, although the expression may be referring to a point entity, it may not be precise with respect to exactly which point is meant. Consider this example:

   (4.6) He called me at *about 3:10 a.m.*

   Here, the expression is of minute granularity, but we do not know which minute is referred to. This is, however, an interpretation issue and does not influence the type of the expression.

2. **Period-referring** expressions: these are references which indicate *how long* something (for example, a process or event) lasted, lasts or will last. Here are some examples:

   (4.7) a. The accounts are paid in full for *the six months ended March 31.*
   b. The festival will last for *5 days starting January 5th.*
   c. *The last 3 days of the battle* were extremely brutal.
   d. The ship was at sea for *two weeks.*
   e. He will make a *three-day* visit to Norway.
   f. This tank can be emptied in *exactly one minute and ten seconds.*

   An alternative term that we will use instead of ‘period’ is **duration**. In cases where the expression indicates when the period started or ended (see the first three sentences above), we will call these expressions **anchored periods** (or **anchored durations**).

We can also consider plural temporal entities, i.e. sets of points and sets of periods, and temporal expressions that refer to these sets. These expressions tell us *how often* something occurs or happens (see Example (4.8a-d)), or indicate *a number of entities* (see Example (4.8e-f)).

(4.8) a. The incident recurred *every month*.
b. The family has visited the cemetery *monthly* since his death.
c. We watch TV *every day*.
d. I go swimming *every four days*.
e. Last summer, I went to the beach on *numerous Saturdays*.
f. He is working on this project *three days a week*.

In the literature these expressions are generally called **set expressions**; sometimes the term **aggregates** is also used.

Following the definitions found in TIMEX2 and TimeML, we require a temporal expression to be a consecutive sequence of text tokens. This may be a sequence of words and numbers, like *five months*, *next week* or *18 July*, or a pattern-based alphanumeric expression of time used in a conventionalised manner, like *12/04/2005* or *12:36am*.

Each temporal expression contains a **trigger**. This is a designated lexical item which, when it occurs in a text, signals the possible presence of a temporal expression. Common triggers are the names of months, weekdays and temporal units. The presence of a trigger in a sentence does not mean that we definitely have a temporal expression; for example, the word *spring* may be a part of a temporal expression, but it may also be used in a non-temporal sense, as in *The watch stopped working because a spring broke*. Expressions that take the form of the conventionalised patterns introduced above are considered triggers; these can be ambiguous just as lexical triggers can be. For example, *1960* may denote a year, but also be a simple number (such as a count of the people that attended a performance in a theatre).

From a syntactic point of view, a temporal expression can be a noun phrase (*five days*), an adjective (*day-long*) or adverb (*lately*), or an adverbial phrase (*at least monthly*); see the sentences in Example (4.9).[6]

(4.9) a. It will take me [*five days* $_{\mathsf{NP}}$].
b. [*Next Tuesday* $_{\mathsf{NP}}$] is supposed to be very hot.
c. I was sick [*yesterday* $_{\mathsf{NP[N]}}$].
d. A [*day-long* $_{\mathsf{JJ}}$] training session is enough to start this job.
e. He always comes late to our [*monthly* $_{\mathsf{JJ}}$] meetings.
f. I have not seen him [*lately* $_{\mathsf{ADVP[RB]}}$].
g. We revise the budget [*monthly* $_{\mathsf{ADVP[RB]}}$].
h. We revise the budget [*at least monthly* $_{\mathsf{ADVP}}$].
i. We revised the budget [*three days ago* $_{\mathsf{ADVP}}$].

Very often the temporal expression is part of a prepositional phrase, as shown in Example (4.10). Following the TIMEX2 guidelines, we do not consider the preposition to be part of the temporal expression, on the grounds that it signals a temporal relation

[6]We use here the Penn Treebank tag set presented by Marcus et al. (1993).

between the time of the event and the reference temporal entity; temporal expressions are the linguistic constructs used to denote these reference entities.[7]

(4.10) a. I was born [in *December* $_{\mathsf{PP}}$].

b. I will call you [before *Monday* $_{\mathsf{PP}}$].

c. See you [at *ten minutes past four* $_{\mathsf{PP}}$].

d. Can you come to me [for *fifteen minutes* $_{\mathsf{PP}}$]?

e. I am going on vacation [for [*two weeks* [*from next Tuesday* $_{\mathsf{PP}}$] $_{\mathsf{NP}}$] $_{\mathsf{PP}}$].

f. I want to receive the reports [on *every Monday* $_{\mathsf{PP}}$] [before *9 am* $_{\mathsf{PP}}$].

Consider also the difference between the last two example sentences above; although both sentences contain two prepositional phrases, in Example (4.10e) we have one temporal expression (*two weeks from next Tuesday*), whereas in Example (4.10f) we have two temporal expressions (*every Monday* and *9 am*). The reason for the different treatment is the syntactic structure: in the former case we have a complex prepositional phrase (which happens to contain another prepositional phrase), but we analyse the latter case as consisting of two prepositional phrases occurring as sisters within the same VP. We take the view that a temporal expression must be a single syntactic constituent, so, although there is a semantic relation between the two PPs in Example (4.10f), we recognize them as two separate temporal expressions.

However, we note that the decision made in TIMEX2 to exclude leading prepositions in all situations overlooks a subtle difference between some cases. Consider the following example:

(4.11) Hurry up! It is *8:24*, the train is leaving in *five minutes*!

When adhering to the TIMEX2 guidelines (Ferro et al., 2005, pp. 24, 27), the second temporal expression in the above example is supposed to be interpreted as denoting time 8:29, which means that what really gets interpreted is *in five minutes*, although the guidelines stipulate annotation of *five minutes* only; however, if we exclude the preposition from the expression, it should be considered as denoting a period lasting five minutes, not a point in time. We also note that including the preposition as part of the temporal expression in this case would be consistent with TIMEX2 considering *three minutes ago* as a temporal expression in the following sentence:

(4.12) What a pity, we missed the train! It left *three minutes ago.*

4.3 Temporal Expressions Referring to Points

We distinguish two major subtypes of point expressions: **explicit** and **indexical**. An explicit expression has a context-independent interpretation, and so the referred-to temporal entity can be marked on a timeline using only the information carried by the expression. Indexical expressions, on the other hand, are context-dependent, and they cannot be fully interpreted—i.e. the referred-to entity cannot be determined—without using additional information from outside the expression. Depending on the way in which the interpretation of the expression relies on the context, we further distinguish four subtypes of indexical expressions, as represented in Figure 4.3.

[7]The notions of the time of the event and the reference time come from the work of Reichenbach (1947).

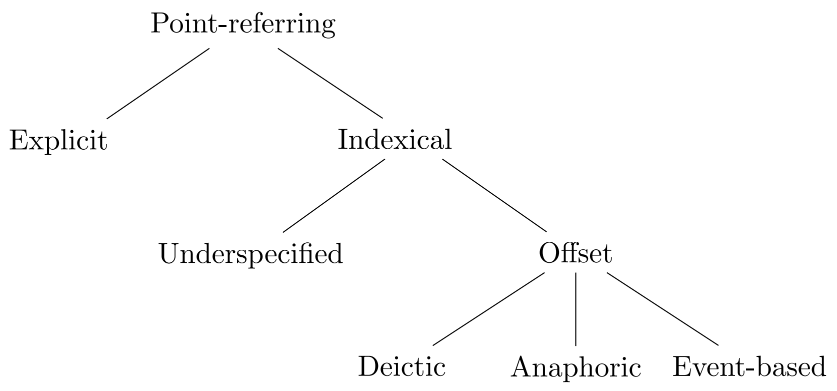


Figure 4.3: A taxonomy of point-referring temporal expressions.

4.3.1 Explicit Expressions

Explicit expressions are those which unambiguously indicate the temporal entity referred to, with no contextual information being required in order to interpret the expression; examples are provided in Example (4.13) below. For this reason we will also call these expressions **fully-specified**.

(4.13) a. I was born in *1980*.

b. I was born in *December 1980*.

c. I was born on *21 December 1980*.

d. I was born on *Sunday 21 December 1980*.

e. I was born on *Monday, 02 January 2006, 6:05pm*.

f. I was born in *the 1980s*.

g. Date and time of last update: *02/01/2006 6:05:31 PM*.

h. Let's meet here again at *four o'clock July 7, 2012*.

i. The first bombing took place on *25 December 2001, twenty minutes after ten in the morning*.

j. There were many military conflicts in *the 20th century*.

k. This is an example of a *12th century* church.

l. This poem was written in *500 BC*.

m. *Summer of 1984* was exceptionally warm in the southern Utopia.

n. We made a huge profit in *fiscal 2005*.

o. We made a huge profit in *the first quarter of FY2005*.

Of course, one could argue that even these expressions are context-dependent because they do not tell us what calendar should be used in their interpretation. We assume in our work the Gregorian calendar as a default, given its status as the internationally-accepted civil calendar. In almost all texts that we are likely to deal with, dates from any other calendar would be explicitly indicated as such, as in the following example:

(4.14) However, in *March 1917* (*February in the pre-revolutionary Russian calendar*), the Czar was overthrown in the February Revolution and the Russian Caucasus Army began to fall apart.

4.3.2 Indexical Expressions

An indexical expression is one whose meaning is partially determined by the context of utterance. We distinguish here two subtypes: **underspecified expressions** and **offset expressions**. The former are explicit expressions with some pieces of information missing, and the latter express a function which returns the temporal location of the entity referred to by the expression.

In both cases determining the actual point referred to by the expression requires finding the contextually-defined **reference time** (also referred to in the literature as the **index time**, hence the name of the type of the expressions). The reference time may be the value specified by the document time-stamp (in which case we call the expression **deictic**), another temporal expression (in which case we have an **anaphoric expression**), or the time-stamp of an event (in those cases where we have **event-based expressions**, discussed further below).

4.3.2.1 Underspecified Expressions

An underspecified expression is similar to an explicit expression, except that it lacks information concerning some temporal units of coarser granularity than the smallest granular unit used in the expression. Example (4.15) presents a number of sentences containing underspecified temporal expressions.

(4.15)
a. I met him in *November.*
b. I met him on *the second of November.*
c. Let's meet again on *the nineteenth.*
d. I met with him at *fifteen minutes to one.*
e. When I entered the room, the clock struck *one.*
f. We need to sell 100,000 copies in *the 4th quarter.*
g. This monument was built in *'63.*
h. This bridge was open in *October '63.*
i. His parents were born in *the '50s.*
j. John visited Marry on *Monday* and told her what we had done.

In each of these cases we are provided with partial temporal information, but because of the missing elements we do not know the exact location of the temporal entity on a timeline. For example, the expression *the second of November* is underspecified because it provides information about the month and day, but does not mention the year. The expression *October '63* is also underspecified, because it does not tell us in which century the 63rd year referred to is located.

Underspecified expressions can be interpreted either deictically or anaphorically, but the expression itself does not indicate which is the case; this is determined by the context in which the expression is used.

4.3.2.2 Offset Expressions

Offset expressions encode a function (hence we will also call these expressions **functional**) which, when applied to the reference time, returns the temporal location of the entity referred to by the expression. This temporal function corresponds to either the addition or subtraction of a number of units at some granularity; for example, *last year* is equivalent to subtracting one year and *three days later* requires adding three days. The temporal expressions themselves indicate what we will refer to as the **direction of offset** (past or future), the number of units, and the unit itself. Sometimes the number does not need to be expressed explicitly. For example, *the following day* implicitly denotes addition of one day. In some cases all the three elements may be encoded in a single word, e.g. *tomorrow*.

For some temporal expressions the offset function is constructed slightly differently. Consider the expressions in Example (4.16).

(4.16) a. I will give a talk *next Monday*.

b. I visited my uncle in Melbourne *last summer*.

c. Let's go to the mountains *this weekend*.

In these cases the addition/subtraction function operates at one level coarser granularity (e.g. week for weekday names, year for year seasons), and then there is an additional selection operation which picks the proper entity within that calendar unit. For example, *next Monday* from Example (4.16a) first requires the identification of the week following the week of the reference time, and then the selection of Monday from that week.

If the reference time is the utterance time (e.g. the time of speaking, or the date of publication), we call the expression deictic; if the reference time is one of the temporal entities previously mentioned in the discourse, we call the expression anaphoric. Finally, the reference time may be the time-stamp of an event mentioned in the discourse; in such case we say the expression is event-based. All three types are exemplified below:

(4.17) a. Hi John! How are you? Can you give me a ride to the shopping centre? My car broke *yesterday* and I won't have it fixed until *next week*.

b. Caroline visited me for dinner <u>*last Saturday*</u>. As this was her first visit at my house, I started the preparation already on *the previous day*.

c. We broke on *the 13th December*. I must admit, I have never been happy again since *the day* <u>*Joanne left*</u>.

The temporal expressions in Example (4.17a) are both deictic; the expression *the previous day* in Example (4.17b) is anaphoric, as its reference time is the value of the underlined expression *last Saturday*; and the expression *the day she left* in Example (4.17c) is relative to the event of Joanne leaving.

Deictic Expressions Deictic expressions are offsets which are naturally linked with the agent making the utterance and her temporal locus. Example (4.18) presents some sentences containing such expressions.

(4.18) a. I met him *yesterday*.

b. I met him *three years ago*.

c. The Jurassic Period began *210 million years ago*.

d. I am going to the USA *next week*.

In the case of written documents, e.g. news stories, the utterance time may be provided via an explicit meta-tag such as the date of publication, or the last date of modification.

Anaphoric Expressions Anaphoric expressions are offsets which need to be interpreted with respect to another expression occurring in the text. Example (4.19) presents a number of such cases.[8] The reference point may occur in the same sentence (see Examples (4.19a)–(4.19d)), in the preceding sentence (Example (4.19e)), or sometimes much earlier in text.

(4.19) a. The first explosion was at *11:14:46 pm* and the second one was *sixty seconds later.*

b. I popped the question in *August 2006* but we got married *two years later.*

c. He called me on *17th January* and he died *the following day.*

d. I received one treatment on *Thursday* and another one *two days earlier.*

e. He died in a car accident in *1978*. He got his driving licence in *the previous year.*

A specific subset of anaphoric expressions are **coreferential** expressions. These are expressions which require nothing to be added to the reference time in order to compute their values; see Example (4.20).

(4.20) a. He graduated in *1999* and got his first job *that same year.*

b. My son was born on *28 December 2001*. I will never forget *this day.*

c. I graduated on *5 July 2004*. I got the job *the same month.*

d. We bought a house in *January 1998*, but we did not move in until *August of that year.*

A coreferential expression may have the same temporal value as the antecedent, as in Examples (4.20a) and (4.20b), be of coarser granularity, as in Example (4.20c), or shift to a different entity within some temporal unit mentioned in the antecedent (e.g. in Example (4.20d) the expression uses only the year information from the antecedent).

Event-based Expressions Event-based expressions identify a temporal entity by means of a reference to an event, as shown in Example (4.21).[9] Determining the actual point in time referred to by an event-based temporal expression requires identifying the underlying event, determining its temporal location, and then calculating the offset. Consider the following examples:

(4.21) a. *Ten seconds after* the second explosion the plane hit the ground.

b. We threw away the Christmas tree *two weeks after* Christmas.

c. At *the time of* the peace agreement the United States agreed to replace equipment on a one-by-one basis.

d. She got a salary raise *the day* he lost his job.

In each of the first two cases we have a non-zero offset from the event; in the third and fourth cases, the referred-to temporal entity is the temporal location of the event, i.e. the offset is zero. Note, however, that while in Example (4.21c) the granularity of the temporal expression is the same as the granularity of the event's time-stamp, in Example (4.21d) the expression explicitly specifies day granularity, while the granularity of the event's time-stamp may be finer.

[8] In each example, the underlined expression serves as a reference time.

[9] We borrow the term 'event-based' from TIMEX2.

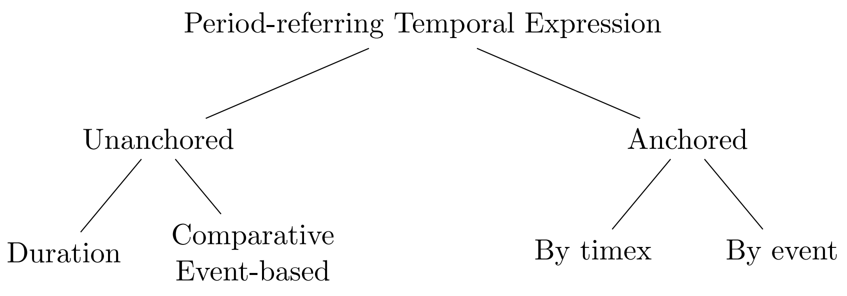


Figure 4.4: A taxonomy of period-referring temporal expressions.

There are also cases where the underlying event is only hypothetical, planned, or possible in the future. Consider Example (4.22); here, the document from which the sentence is sourced does not specify the time of Germany's defeat because it is referred to as a hypothesized future event.

(4.22) The conference determined that the Soviet Union would declare war on Japan within *three months of* <u>*Germany's defeat*</u>.

Special attention is required in the case of birthdays and anniversaries. These are in some sense events; however, they are directly linked with temporal entities, and therefore their mentions in a text may be considered to be temporal expressions.[10] Depending on the context of their use, such temporal expressions may be used anaphorically or coreferentially. Consider the following two examples:

(4.23) a. I was born on <u>*16 June 1980*</u>. On *my 30th birthday* we hired a limo to celebrate it in a special way.

b. <u>*This Wednesday*</u> is *my 30th birthday.*

In Example (4.23a), the expression *my 30th birthday* indicates that an offset of thirty years is to be calculated from the date of the antecedent; in Example (4.23b), the same expression is coreferential with the expression *This Wednesday.*

4.4 Temporal Expressions Referring to Periods

Every period has some duration. If a period can be placed on a timeline, then we say it is **anchored**; otherwise all we have is the duration, which we also call an **unanchored** period. This distinction corresponds to the top-level division in our taxonomy concerning period-referring temporal expressions. A number of finer distinctions are also made, as shown in Figure 4.4.

4.4.1 Unanchored Periods

Temporal expressions that correspond to unanchored periods are those which do not provide information about the beginning or end of the period; they only state the length of the period (i.e. its duration). This may either be because the expression

[10]TIMEX2, for example, treats the words *birthday* and *anniversary* as triggers for temporal expressions.

is part of an utterance stating a generalization (see Example (4.24)), or the temporal location is simply not stated within the extent of the expression (see Example (4.25)), but may be indicated elsewhere or assumed known.

(4.24) a. Cleaning one office on this floor takes *30 minutes.*

b. It takes *365.25 days* for the Earth to rotate around the Sun.

(4.25) a. The Nile Movie Festival lasted *five days.*

b. We will be traveling across the country for *two months.*

An expression of duration can mix temporal units of different granularities, as in Example (4.26):

(4.26) a. This project will run for *one year and two months.*

b. The current men's record in the marathon is *two hours, three minutes and fifty nine seconds.*

c. Both cycling and driving from the beach to the city center takes *two hours and forty minutes.*

A duration does not need to be expressed directly by a number and unit. It may also be presented by comparison to another explicitly-stated duration, or to the duration of some described eventuality. We call the latter type of expression a reference to a **comparative event-based duration**. Consider Example (4.27):

(4.27) a. Our next meeting will be *15 minutes shorter than the last interview.*

b. This marathon record is *32 seconds better than the previous record.*

The duration associated with the eventuality is required in order to calculate the duration of the temporal entity referred to by the temporal expression as a whole. Of course these expressions can also express generalizations:

(4.28) Walking to the shopping center takes *ten minutes longer than going by bus.*

or mix different granularities:

(4.29) Walking to the shopping center takes *ten minutes and five seconds longer than going by bus.*

4.4.2 Anchored Periods

Anchored periods differ from unanchored periods in that they specify a starting or ending point. In some cases it is clear directly from the expression which end point is meant (see Examples (4.30a)–(4.30b); in other cases some sort of reasoning must be carried out (compare Example (4.30c)).

(4.30) a. The accounts are paid in full for *the six months ended March 31.*

b. The renovations will last *five days starting tomorrow.*

c. Our company managed to double its profit in *the last four months of 2008.*

It happens quite often that information about the duration and the end points is mentioned in different parts of a sentence, or even in different sentences. Since we require a temporal expression to be a continuous sequence of tokens, we consider such expressions of duration to be unanchored, even though the anchor may be recovered from somewhere else in the text.

The anchor found within the extent of the expression can be a point-referring expression of any type. We demonstrate this in Example (4.31), where we have, respectively, a fully-specified, an underspecified, an offset and an event-based point being used to anchor the same duration.

(4.31) a. The festival will last for *five days starting 5th January 2010.*

b. The festival will last for *five days starting January 5th.*

c. The festival will last for *five days starting next Wednesday.*

d. The festival will last for *five days starting the day when soccer championships finish.*

Some periods may be anchored **implicitly**, which is the case with deictic expressions. Example (4.32) presents two cases where the anchor is the time of utterance.

(4.32) a. *The next three days* will be extremely hot and humid.

b. I will be staying here over *the next two and a half weeks.*

c. I have not checked my email in *the last 72 hours.*

Specific keywords found in the extent of the expression (e.g. *next*, *last*) tell us whether the time of utterance is the starting or ending anchor.

Anchoring information may also be provided by means of an event mentioned within the extent of a temporal expression. In such cases, we call the expression **event-anchored** or **event-based**; see Example (4.33).

(4.33) a. The rate of US combat deaths in Baghdad nearly doubled in *the first seven weeks of the "surge" in security activity.*

b. *The last three days of the battle* were extremely brutal.

c. *The last days of this battle* signified the end of mobile warfare in the west.

d. I was so panicked I could not make a single step for *30 minutes after the earth quake.*

e. There was no terrorist warning in *the three years before the bombing in the Underground.*

f. In *the ten years since the last flooding*, we had no problems with high water.

We consider general references to the past, present and future to be anchored durations. In most cases these are deictic expressions, anchored to the time of utterance; see Example (4.34).

(4.34) a. His performance has deteriorated *recently.*

b. Our company is undergoing many changes *these days.*

c. *The future of our business* depends on what we decide in this meeting.

Note, however, that these expressions can also be anchored anaphorically, as in Example (4.35):

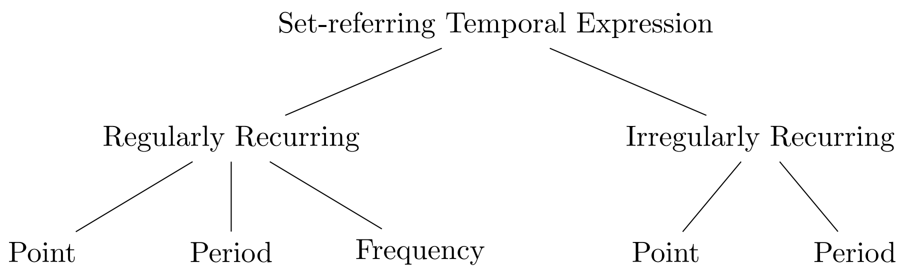


Figure 4.5: A taxonomy of set-referring temporal expressions.

(4.35) a. In 1968 he decided to stop using a car, train, bus etc. to see how it was to live in *the past.*

b. As in the 18th century people did not know electricity, at *those times* people used candles to light their homes.

c. In 1945 many people just had to forget what had happened and had to build *a new future* from scratch.

It is also possible to anchor these expressions using events:

(4.36) *The post-catastrophe future of security regulations* is still a subject of many discussions.

4.5 Temporal Expressions Referring to Sets

A set expression refers to a number of temporal entities. Figure 4.5 shows the different kinds of set expressions we consider. The top-level distinction we draw is between **regularly recurring** and **irregularly recurring** temporal entities; in both cases the referred-to entities may be points or periods.

Independently of what the type of the expression is, a set expression may indicate the **boundaries of the recurrence**, specifying the time frame in which the recurrence takes place. In the example below, there are no boundaries, and we cannot be sure which winters are actually meant:

(4.37) I love snow and sport so much that I go skiing *every winter.*

In the following two sentences, we have one boundary mentioned:

(4.38) a. The tutor will be coming *every Monday from 19 July.*

b. I will continue to go to the gym on *every Tuesday and Friday until the end of the year.*

In the first example, the expression provides the **beginning boundary** of the recurrence; in the second example, the expression specifies the **end boundary** of the recurrence. In the following sentence we have an expression which refers to a set of entities whose occurrence is bounded at both ends:

(4.39) We spent *every winter in the 80s* skiing, but since the accident I haven't skied anymore.

This example shows that the boundaries do not have to be mentioned explicitly.

An expression that specifies two boundaries will be referred to as **bounded**; if only one boundary is provided, we will refer to the expression as **partially-bounded**; and if none of the boundaries is given, as in Example (4.37), then we say the expression is **unbounded**.

4.5.1 Regularly Recurring Temporal Entities

We can use a temporal expression to refer to a set of entities where there is some regularity in regard to the repeated occurrence of those entities over a period of time. The entities may be either points or periods. Example (4.40) provides examples of sentences that contain set expressions referring to points:

(4.40)
- a. This area is flooded *every July.*
- b. I meet with my supervisor *every Monday.*
- c. We used to go camping *every weekend in summer 2002.*
- d. On *Saturday nights* Sue changes her smart office suit for a mini-skirt.
- e. My grandma has to take pills *each morning.*
- f. Because of the crisis in the market, we discuss our situation *weekly.*
- g. Have we already received the *daily* reports from the agency?

Example (4.41) demonstrates set expressions referring to regularly recurring periods:

(4.41)
- a. We spend *the last two days of every month* on summarizing our achievements.
- b. It takes *the first 25 minutes of each shift* for a crane operator to walk up the stairs.

In some cases, it may not be clear whether the expression refers to periods or points; consider Example (4.42).

(4.42)
- a. To keep such a great body she exercises for *three hours every day.*
- b. I work for John *three days every week.*

In these instances, we have no information as to whether the three hours and three days referred to occur consecutively, or whether they are made up of many smaller chunks across a longer period of time. In such cases, we analyse the expression without taking the context into account: here, *three hours* and *three days* are then considered as periods, so we view these set expressions as referring to recurring periods.

We also treat expressions of **frequencies** (or *rates of occurrence*), such as those in Example (4.43), as expressions of sets of regularly recurring times. The intuition here is that the meaning of such expressions implicitly assumes that something is done repeatedly at a specified rate over some stretch of time; i.e. the activity is repeated in chunks of time specified as the temporal unit used in the expression.

(4.43)
- a. The wardens check the building *three times a night.*
- b. I go to a dentist *twice a year.*

A different type of rate is expressed in Example (4.44):

(4.44)
- a. In many urban areas you cannot drive more than 50 kilometres *per hour.*

b. This factory can produce 100 cars *a day.*

c. The rides have been known to draw 350,000 passengers *a year.*

Expressions like these tell us how much is done (e.g. travelled, produced, carried, or achieved) during a single instance of a temporal unit, e.g. an hour, a day, or a year.[11]

4.5.2 Irregularly Recurring Temporal Entities

It is also possible to refer to sets of temporal entities in situations where it is not clear exactly which entities are meant; in particular, we cannot locate them on a timeline, although there may be some constraints expressed, such as the boundaries of the occurrences, or other details that constrain the possible interpretations, such as a specification of the name of the day of the week on which the events referred to take place. Consider Example (4.45):

(4.45) a. Last year, I overslept on *many Monday mornings.*

b. On *some days* my temperature was so high I could not go to work.

c. We visited aunt Mary *twice in 1999.*

d. We have played golf *three times in the past five years.*

e. On *some Saturday nights* Sue changes her smart office suit for a mini-skirt.

f. We spend *the last two days of almost every month* on summarizing our achievements.

In particular, compare Example (4.45e) to Example (4.40d): the change from universal to existential quantification makes it impossible to determine how many and which Saturday nights are being referred to; specifically, we cannot tell whether or not the temporal entity in question is a regular recurrence.

Example (4.45f) presents an expression referring to irregularly recurring periods.

4.5.3 What Counts as a Set?

In some cases it may be difficult to distinguish between a set-referring temporal expression and multiple occurrences of point temporal expressions. Compare the following two sentences:

(4.46) a. I was sick on Monday and Tuesday.

b. I was sick on *the first two days of this week.*

Semantically, both sentences appear to say the same thing in different ways. While there is no confusion about the annotation of the temporal expression in the second sentence, the question arises whether in the first sentence we have two point temporal expressions (*Monday* and *Tuesday*), or one set-referring temporal expression *Monday*

[11]Treating rate expressions like *a day* as set expressions follows the approach taken in TIMEX2 and TimeML. However, they could also be treated as durations (i.e. unanchored periods), rather than as sets; this would be consistent with treating a reference to more than one temporal unit as a duration. For example, consider *four weeks* in *During our round-the-world expedition, on average we travelled 10,000 km in four weeks*). Ultimately, the semantics of set expressions is rather complex, and beyond the scope of what we can reasonably consider here. In general, we follow the stand taken in TIMEX2 and TimeML.

and Tuesday. If we take the latter course, then we would need to introduce a third subtype of set-referring temporal expressions, which do not refer to recurring events.

To a very limited extent, this issue is addressed in the TIMEX2 annotation guidelines (Ferro et al., 2005, Sec. 5.2.3). There, the proposal is to treat all elements of conjunctions or disjunctions as separate temporal expressions. However, the example given in the guidelines, repeated here as Example (4.47), uses conjunction in a manner which does not appear to involve a set.

(4.47) The bug will get fixed between *now* and *Monday morning*.

Here, it makes sense to take the view that we have references to two individual temporal entities. It is not so clear that this is the case in Example (4.46a), however, and the TIMEX2 guidelines do not provide a useful clarification for such cases.

The conclusion from these observations is that it is not possible to treat all instances of conjoined temporal expressions in the same way; whether or not the conjunction describes a set depends on the semantics of the sentence as a whole. In Example (4.46), from a semantic point of view we would want to say that *Monday and Tuesday* is a single temporal expression referring to a set of entities; however, for the purposes of evaluation, in this work we will adopt the simplified approach taken by TIMEX2, and annotate each conjunct as an individual expression referring to a single entity.

We also note that in TIMEX2 there are similar confusions between set expressions and period expressions; for example, *the past three summers*, in our view, refers to a set of three summers, not a period the duration of which is three summers, as is proposed in TIMEX2 (Ferro et al., 2005, p. 31).

4.6 Non-specific Expressions

Some temporal expressions may be **non-specific**, in the sense that they refer to temporal entities that cannot be placed on a timeline. We distinguish two types of non-specificity: **generic** and **indefinite**. In the first case, the expression is typically part of a generally-holding statement or belief, or a description of a universal law or state; e.g. *February is the shortest of all the months*. In the second case, the expression refers to an entity which is linked to some eventuality that takes place in a specificied time but its temporal location cannot or is not supposed to be determined, e.g. *I was born on a Sunday morning*.

Non-specificity concerns expressions of all top-level types; however, we take the view that if a 'point-like' expression is used generically then it automatically becomes a set expression. This is because such expressions refer to a whole class of temporal entities, not a single entity, as shown in Example (4.48).

(4.48) a. In the southern hemisphere *days* are much longer in *January* than in *July*.

b. *Summer* is warmer than *winter*.

In these examples *January*, *summer*, and so on do not refer to any specific month or year season. The singular form may initially suggest that a specific point is meant, but the analysis of the rest of the sentence reveals the generic nature of the utterance.

In Section 4.5 we presented the following example of an unbounded set expression:

(4.49) I love snow and sport so much that I go skiing *every winter*.

Of course, one has to realise that the eventuality that the expression is linked to naturally provides some boundaries which limit the range of instances of the referred temporal entities; here, such boundaries are set by the lifetime of the subject. If we consider Example (4.50), in practice the limits are placed here by the duration of a given project or the period of time for which someone works at a given company.

(4.50) We meet *every Monday* to present new results.

Similarly, we need to accept that generic expressions as in Example (4.48) are also in some way bounded by nature, e.g. the lifetime of the Earth or the existence of a given calendar system. In consequence, an attempt to draw a borderline between generic and unbounded set expressions becomes problematic.

Generic expressions may also be of the period type, as in Example (4.51), and in such cases they are unanchored durations.

(4.51) a. It takes *365.25 days* for the Earth to rotate around the Sun.

b. On average, the EKG test lasts *15 minutes.*

Also some expressions referring to general past, present and future can be non-specific:

(4.52) a. The *present* state is always the outcome of *past* actions.

b. *The future of a relationship* depends on both sides.

An indefinite point expression refers to a temporal point, but the reference does not reveal which point is meant. Note the similarity of these expressions to underspecified expressions, with the important difference being that in the case of indefinite expressions the missing bits of information are not to be found in the interpretation stage, because it is not the intention of the text to refer to the entity unambiguously. Example (4.53) presents a couple of such cases.

(4.53) a. I was born on *a Sunday.*

b. I met my wife on *a sunny day in July.*

Indefinite period expressions also exist, and we distinguish two reasons to consider an expression to be of such a type:

1. No anchor information is available, as in

   (4.54) It took me *five days* to paint this fence.

2. The exact duration is unknown (vague duration), as in

   (4.55) The researchers obtained the drug, but it lasted only *seconds* before it decomposed.

   (4.56) It would take me *a few days* to paint this fence.

The two properties may be co-present:

(4.57) It took me *a few days* to paint this fence.

Vague duration may also appear for anchored period expressions (here, the duration is anchored deictically):

(4.58) I have been repairing this car for *the last few days.*

All the above sentences contain indefinite period expressions.

Finally, Example (4.59) presents indefinite set expressions.

(4.59) a. I attended lectures only on *some Mondays in 2004*.

b. I called my mom *every three days in 1999*.

c. I called my mom *every few days in 1999*.

d. We visited aunt Mary *twice in 1999*.

All these expressions are bounded; in our view, there is no unbounded set expression that would be indefinite (as noted earlier, we assume such expressions to always be generic). Note also that Example (4.59c) is not only indefinite but also vague.

4.7 Conclusions

In this chapter we have presented a taxonomisation of temporal expressions. The taxonomy was developed based on the more limited taxonomies described elsewhere in the literature, the example temporal expressions found in the TIMEX2 guidelines, and temporal expressions found in real texts including those in the ACE 2005 Training corpus and our WikiWars corpus. The taxonomy shows the diversity of types of references to temporal entities and systemises the types in a hierarchy. It serves as a starting point for understanding the range of problems that are involved in the interpretation of temporal expressions.

We drew a top-level distinction between temporal expressions referring to single and multiple temporal entities, and between point and period forms of these entities.

We divided single-point referring temporal expressions into those which are context-independent (explicit) and those which are context-dependent (indexical); we further divided the latter into a number of subtypes.

In the case of period referring expressions, we distinguished anchored and unanchored periods. The anchor of a period is one of the end points of the period that must be specified within the extent of the expression, and it can be provided either by means of a temporal value (i.e. there is a subexpression referring to a point) or by means of a reference to an event.

The semantics of set expressions is very complex, and we have only touched upon some of the issues here. We distinguished references to regularly and irregularly recurring entities; we also indicated that there are cases of conjoined expressions which should be classified as non-recurring set-referring expressions, rather than as multiple instances of point temporal expression as is the case in TIMEX2.

Finally, we discussed non-specific expressions, which are expressions that cannot be placed on a timeline. These, however, do not constitute a separate type in the taxonomy; rather, non-specificity is a semantic feature that may be true of various types of temporal expressions.

The taxonomy presented here provides a more detailed analysis of the range of temporal expressions than is found in the existing literature (cf. (Mani et al., 2001; Schilder, 2004; Ferro et al., 2005; Saquete, 2005; Han et al., 2006a)). It also addresses some problems with the other accounts (e.g. we treat underspecified expressions as indexicals, not explicit expressions). A reader might be curious as to the relation between our taxonomy and the TIMEX2 scheme, which we use in our implementations.

TIMEX2 focuses on the description of the values that need to be provided in the annotations, and not on the distinguishable types of expressions; this is, in a way, reflected in the absence of an attribute in the scheme that corresponds to the notion of type. Of course, the guidelines implicitly provide some level of categorisation of temporal expressions, otherwise there would be no means of telling when to use a particular annotation value format. However, if one prepared a taxonomy of the expressions based on the TIMEX2 guidelines, it would be less detailed and less hierarchical than our proposal (the guidelines are not concerned with capturing the relations between the types). In some cases we presented alternative views to those found in the guidelines, but for the purposes of evaluation we will need to adopt TIMEX2. Still, the distinctions we have drawn will be of importance in analysing the problems involved in the interpretation of temporal expressions, which we turn to next.

Chapter 5

Extent Recognition

The first step in the extraction of temporal expressions from text is the proper recognition of their occurrence. This can be further divided into two phases: **detection** and **extent recognition**. Detection is about 'spotting' the existence of an expression, which can be achieved, for example, by finding time-related words (tokens) such as weekday names.[1] The goal of extent recognition is to precisely determine where the expression starts and ends, i.e. to establish which tokens make up the expression. It is quite common to evaluate a temporal expression tagger separately on the detection (lenient results) and extent recognition (strict results) tasks.

Very often these two tasks are based on so-called **trigger**-based approaches.[2] A trigger is a lexical item whose presence is a strong indicator that there may be an instance of a temporal expression; for example, names of months, weekdays and temporal units are triggers. Identification of a trigger which is in fact part of a temporal expression means that we have detected a temporal expression; making an annotation around a trigger word which is not actually part of a temporal expression (for example, as in the word *March* in *The Long March*, which is the name given to a famous military retreat) generates a false positive, i.e. a spurious match. Recognition then extends the span of a potential temporal expression beyond the trigger word itself. In some cases a trigger may constitute a temporal expression on its own, as in Example (5.1) below, but in many cases, as in Example (5.2) (where we have distinguished the triggers from the rest of the temporal expressions that contain them using bold face), a trigger may be syntactically modified and be only one of several tokens that make up the temporal expression.

(5.1) a. I did not go to work on ***Monday***.

b. My son was born in ***1995***.

c. I was woken up by the fireworks at ***midnight***.

(5.2) a. I met with Joanne *two* ***weeks*** *ago*.

[1]In some evaluation schemes, e.g. the ACE TERN 2004 evaluations, it is even enough to identify a single character as belonging to a temporal expression.

[2]There are two other approaches that do not involve identification of a trigger. One, known from named entity recognition and undertaken in the context of temporal expressions by Hacioglu et al. (2005), is to carry out classification using the B-I-O model, i.e. each token is classified as being a Border (the first or last) token of a temporal expression, an Inside token (between two border tokens), or Outside the expression. Another approach, proposed by Ahn et al. (2007), is to use a machine-learning classifier which for each syntactic constituent that could be a temporal expression (e.g. any NP or ADVP) decides whether it is a temporal expression or not.

b. I spent *three and a half* ***months*** in Spain.

c. Mary was promoted *the* ***day*** *her husband lost a job.*

In the majority of the systems presented in the literature (see, for example, (Mani and Wilson, 2000b), (Schilder, 2004), (Negri and Marseglia, 2005)), the remaining part of the extent is determined by means of hand-written recognition grammars[3] which consist of a number of detailed rules built around finding a trigger in an appropriate context; with such an approach, a knowledge engineer must predict the range of temporal expressions that can appear in text, for the benefit of getting more control over the system's behaviour and not being dependent on large volumes of training data.

As an alternative to these finite-state pattern-based approaches, we can observe that temporal expressions are syntactic constituents; given the trigger as a seed, we can then grow the extent outwards to see if we can find the complete temporal expression as a syntactically-determined element. There are two ways we might do this: using functional dependencies, or using syntactic constituency. If we assume that the trigger is the syntactic head of the constituent corresponding to the temporal expression, which is what TIMEX2 (Ferro et al., 2005, p. 7) stipulates in its definition of a markable expression, the use of a dependency-based approach seems straightforward: we only have to look down the dependency subtree rooted by the trigger to the ends of the branches. When using syntactic constituency, we find in the tree a leaf that is a trigger and we select as the extent of the temporal expression a syntactic constituent containing this leaf. Perhaps surprisingly, neither of these approaches has so far been presented in the literature. In this chapter we present experiments using these two methods; in both cases we use four syntactic parsers: Minipar, Connexor and C&C provide us with dependency trees, Bikel/Collins, Charniak and Charniak-Johnson parsers produce syntactic constituency trees, and the Stanford parser is used for both types of syntactic information.

[3]This is true of 19 out of the 27 tagging systems which we listed in Chapter 2.

5.1 The Extent of Temporal Expressions

Generally speaking, temporal entities (i.e., those entities that are the referents of temporal expressions) are arguments to propositions. This means that in sentences they are located, as are named entities more generally, as the syntactic arguments or adjuncts to verbs.[4] This might suggest that temporal expressions are typically either noun phrases (NPs), prepositional phrases (PPs) or adverbial phrases (ADVPs). However, for the purposes of evaluation, we adopt the TIMEX2 characterisation of temporal expressions; under this view, in a prepositional phrase that expresses temporal information, it is the constituent NP that is the temporal expression, and the preposition is not considered part of the temporal expression.

In the TIMEX2 scheme, it is also the case that some temporal expressions function as modifiers in noun phrases, e.g. *Monday* in *the Monday meeting.* The general approach in TIMEX2 is to include in the extent the temporal trigger and all its syntactic pre- and postmodifiers. But some special structures are dealt with separately; for example, in the case of range expressions (see Example (5.3)) and conjoined expressions (see Example (5.4)) there are two temporal expressions to be recognized, although there is only one trigger.

(5.3) a. Dinner is from *five* to *six pm* ***tomorrow***.

b. It usually takes *five* to *six* ***years*** to complete a project of this scope.

(5.4) Britain is staying outside the currency union for *at least the next* ***year*** or *two.*

In particular, a text string may contain:

(5.5) a. <u>a single temporal expression</u>:
The concert is at [*8:00 p.m. Friday*].
[*The summer of this year*] was unusually hot.

b. <u>multiple temporal expressions without embedding</u>:
The concert is at [*8:00 p.m.*] on [*Friday*].
I tutored an English student [*some Thursdays*] in [*1998*].

c. <u>multiple temporal expressions with embedding (nested expressions)</u>:
[[*This year*]*'s summer*] was unusually hot.
I'm leaving on vacation [*two weeks from* [*next Tuesday*]].

We might dispute some of the stipulations that TIMEX2 makes, especially in the light of their consequences; for example, *This year's summer* and *The summer of this year* mean the same, but TIMEX2 considers them to contain different numbers of temporal expressions, even despite the fact that in both cases there are two triggers: *year* and *summer.*[5]

As noted above, TIMEX2 does not consider a leading preposition to be a part of a temporal expression; but an exception is made to this rule in frequency (rate) expressions, as in Example (5.6).

(5.6) The machine produces on average 3.14 items *per hour.*

[4]Verbs, on the other hand, express eventualities, which also convey temporal information, but of a kind that is beyond the scope of the present work.

[5]By way of contrast, in TimeML no nesting of annotations is allowed and *This year's summer* is considered to be a single temporal expression.

This might be considered an inconsistency.

There are other aspects of the annotation scheme we might question; for example, given the guidelines for the annotation of range expressions (Ferro et al., 2005, p. 59), the string *the night of 13 February–14 February* needs to have two temporal expressions annotated as in Example (5.7a).

(5.7) a. 13 cars were stolen in the city during
[the **night** of 13 **February**]–[14 **February**]. TIMEX2

b. 13 cars were stolen in the city during
[the **night** of [[13 **February**]–[14 **February**]]]. An alternative

We note, however, that the string contains three temporal triggers (*night* and two occurrences of *February*); therefore, an alternative solution is presented in Example (5.7b). Here, we have two temporal expressions referring to the dates and one larger expression referring to the night that spans the two days.

The purpose of highlighting these observations is not primarily to criticise TIMEX2, but to indicate that the issue of defining the extent of a temporal expression (and also in the first place what a temporal expression is) is more complex than it might seem at first. Despite our reservations about some aspects of the TIMEX2 scheme, the lack of relevant resources tagged using any other scheme means that for the purposes of evaluation we use TIMEX2-based data along with its possibly less-than-optimal annotation choices.

5.2 Syntactic Information for Extent Recognition

The development of a pattern-based grammar for the broad-coverage recognition of temporal expressions is a laborious task, made further complicated by the requirement that the knowledge engineer needs to explicitly control interactions between rules. With such rules, usually the temporal expressions are recognized by matching specific lexical items; some generalisation may be achieved by using POS tags. In such an approach, as the grammar grows in size, it becomes difficult to provide broad coverage and at the same time to keep the precision at a high level.

Given the observation that temporal expressions, within the currently adopted definitions, are in fact syntactic constituents of sentences, we might attempt to obtain a higher level of generalisation by using syntactic information. The hope here is that, instead of having to predict what sequences of individual words can constitute temporal expressions, we might be able to develop a conceptually much simpler algorithm that, provided we have a syntactic analysis of a sentence, would choose those constituents which correspond to temporal expressions.

An aspect of this idea has been investigated by Ahn et al. (2007), who used a machine-learned binary classifier (an SVM) which, for each phrase in a sentence that is potentially a temporal expression (e.g. NP, PP or ADVP), decides whether or not it actually is.

However, we take a slightly different approach. Ahn et al. (2007) tried to learn what makes a given constituent a temporal expression. We assume we have already identified the head of a temporal expression via its trigger, and we try to determine the full extent of the expression built around this trigger.

Ahn et al. (2005a) used shallow (chunk) parsing; this provides a sequence of non-overlapping units which are effectively approximations to phrases. Ahn et al.'s recognition patterns looked for chunks headed by trigger words and, if necessary, joined them with a required complement chunk. For example, the expression *24 years after his death* would be recognized with a pattern corresponding to a noun chunk (*24 years*) followed by a complement preposition chunk (*after his death*). The results achieved were, however, lower than those obtained with their corresponding grammar that did not use chunks. Ahn et al. (2005a) provide some examples of expressions that are not recognized by their grammar, and claim that a deeper syntactic analysis is required to successfully process such cases. We developed our own chunk-based recognizer that contained more sophisticated patterns for recognizing those expressions that Ahn et al. found problematic (such as that in Example (5.8a)) and some other syntactically-complex expressions we found in the WikiWars corpus (such as those in Examples (5.8b) and (5.8c)).

(5.8) a. two years after the crash site was discovered
b. two days after his arrival in Jerusalem
c. the same time as the battle in Poland

However, such event-based descriptions can be arbitrarily complex, and so we are drawn to the same conclusions reached by Ahn et al. While it is possible to add more chunks to the extent or to define a pattern with a Kleene star operator denoting any number of occurrences of an element, in doing so we might include PPs which are not attached to the temporal expression but to the verb chunk; this problem occurred, for example, when processing the sentence 'Hundreds of Iraqi civilians and police were killed over *the next few months* in a series of massive bombings', where the chunk-based approach included in the extent the PP *in a series of massive bombings*. There are also problems when a single chunk may contain two temporal expressions (e.g. *the spring and summer*) and when a temporal expression functions as a modifier (e.g. *daily* in *initial daily reports*), in which case the whole chunk is annotated, rather than just the part that corresponds to the temporal expression.

So, we need to turn to more sophisticated means that give us more detailed syntactic information. We can distinguish two popular types of syntactic analysis based on the type of information they output; this may be either a phrase structure which provides the syntactic constituency of a sentence, or a set of functional dependencies between the tokens in the sentence.[6] Consider Example (5.9).

(5.9) He returned some gifts *five days after Christmas*.

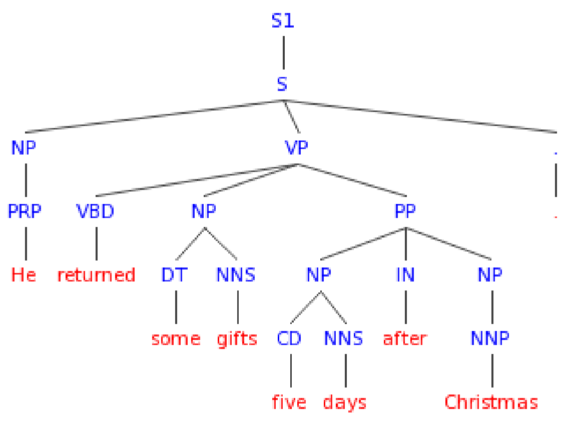
Phrase structure can be presented using a labelled bracket notation; see Figure 5.1a for the representation corresponding to the above example. A tree of syntactic constituents, such as that in Figure 5.1b, presents the same information in a graphical way. A functional dependency parser is a parser which finds functional dependencies between the tokens in a sentence. Usually it also assigns morphosyntactic tags to all tokens. A functional dependency is a relation (which in some frameworks may be named) between two tokens, one of which is dependent on the other, with the latter being referred to as the head.[7] A dependency can be expressed in the form `depFunName(head_id, dep_token_id)`; alternatively, some parsers, for a given token, provide the name of the dependency and the ID of the head token, which can be notated

[6]There exist algorithms that, for some dependency frameworks, can convert dependency structures to phrase structures and vice versa; see, for example, (Xia and Palmer, 2001).

[7]In the literature we also find terminology where the relation holds between a child and a head, or a modifier and a modifiee.

```
[S1
 [S
  [NP [PRP [ He ] ] ]
  [VP
   [VBD [ returned ] ]
   [NP [DT [ some ] ] [NNS [ gifts ] ] ]
   [PP
    [NP [CD [ five ] ] [NNS [ days ] ] ]
    [IN [ after ] ]
    [NP [NNP [ Christmas ] ] ]
   ]
  ]
  [ . [ . ] ]
 ]
]
```

(a)



(b)

Figure 5.1: An example output of a phrase structure syntactic parser presented by means of (a) labelled bracket notation and (b) a tree of syntactic constituents (b).

as follows: `depFunName:>head_id`. Figure 5.2a presents the results from a dependency parser applied to Example (5.9). As each token can have only one head, it is possible to present the output as a tree as in Figure 5.2b. The results from two dependency parsers may be quite different, reflected in different sets of dependency types[8] and different treatments of phenomena such as coordination (as experienced by Clark and Curran (2007, p. 536) in the evaluation of their C&C parser). The conversion between two different dependency representations is not always straightforward; the issues have been broadly discussed in the literature concerned with the evaluation of parsers (see, for example, (Carroll et al., 1998), (Crouch et al., 2002), (Preiss, 2003), (Clark and Curran, 2007, Sec. 11)).

It seems plausible that both phrase structure and functional dependencies could be used to recognize temporal expressions. Once a temporal expression is detected by finding a temporal trigger, the extent is then determined either by selecting a relevant phrase from the superordinate structure of syntactic constituents, or by taking the extent of the subtree in the dependency tree which has the trigger as root. Referring again to Example (5.9), the relevant constituent phrase is the whole PP (see Figure 5.1b), and in the case of using a dependency tree the extent is determined by the tokens in the subtree of *days* (see Figure 5.2b). In theory, this approach should also work for more complex cases, when a temporal expression includes a dependent clause, as in the following example:

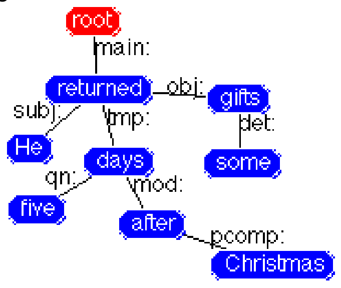
(5.10) I remember *the* ***days*** *when he was the best in the team.*

The corresponding syntactic and dependency trees for this sentence are presented in Figure 5.3. We can see that in both cases using a relevant subtree would result in the correct extent of the temporal expression, i.e. *the days when he was the best in the team.*

[8]For example, Carroll et al. (1998) for their parser evaluation scheme proposed a set of 20 grammatical relations, but in the Stanford parser (de Marneffe et al., 2006), which used that work as a backbone, there are already 48 grammatical relations distinguished; the authors motivate their extensions on the basis of their significance for real-world applications, rather than theoretical concerns.

```
1  He        he        subj:>2  @SUBJ %NH PRON PERS NOM SG3
2  returned  return    main:>0  @+FMAINV %VA V PAST
3  some      some      det:>4   @DN> %>N DET
4  gifts     gift      obj:>2   @OBJ %NH N NOM PL
5  five      five      qn:>6    @QN> %>N NUM CARD
6  days      day       tmp:>2   @ADVL %EH N NOM PL
7  after     after     mod:>6   @ADVL %EH PREP
8  Christmas christmas pcomp:>7 @<P %NH N NOM SG
9  .         .
10 <s>       <s>
```

(a)



(b)

Figure 5.2: An example of results of a functional dependency parser presented by means of a list of dependencies (a) and a corresponding dependency tree (b).

In what follows, we experiment with generic algorithms using dependency and constituent analyses of sentences. By augmenting the generic approaches it should be then possible to adjust to the sometimes idiosyncratic requirements of the TIMEX2 specifications. In this sense our algorithms provide first-cut approaches to the problem of extent recognition using syntactic information.

5.3 The Selection of Triggers

The approach we propose depends on the prior identification of a trigger term. The literature does not provide an agreed-upon or recommended list of triggers (not even the TIMEX2 annotation guidelines provide an exhaustive listing); so, we need to first derive a set of triggers.

The point of the process for building a list of triggers described here is to enable detection of a broad-enough range of temporal expressions to test the proposed recognition methods; it is *not* a test of our ability to detect triggers. We want to see what difference in results are obtained by using syntactically-derived extents; we therefore need a reasonably broad sample of triggers, so that we do not accidentally miss expressions that have interesting syntax just because we omitted the trigger terms they contain. For this experiment the absolute values of the lenient and strict results do not matter that much; it is the difference between them that we care about. The only caveat is that if the lenient score is low, then this suggests that we may not have captured the full variety of triggers in the data.

We use a semi-automatic procedure to build our list of triggers. We analyse the annotated corpora used in the experiment, i.e. the ACE 2005 Training corpus and our in-house developed WikiWars corpus (see Chapter 3), to identify the words that appear in the extents of the gold-standard annotations, along with their frequencies; based on the resulting list we then decide what triggers to use. We first run the Alternate Tokeniser available in the GATE framework to tokenise the texts; then for each lexical form occurring in the corpora we count how many times it appears within a temporal expression, and how many times it appears outside a temporal expression. We remove from the candidate list any tokens that do not occur within any TIMEX2 annotation, since obviously these cannot be triggers; this leaves us with 1489 unique forms. From these, we first consider those terms that appear only within temporal expressions; this is a set of 387 unique forms. We manually remove those that, on

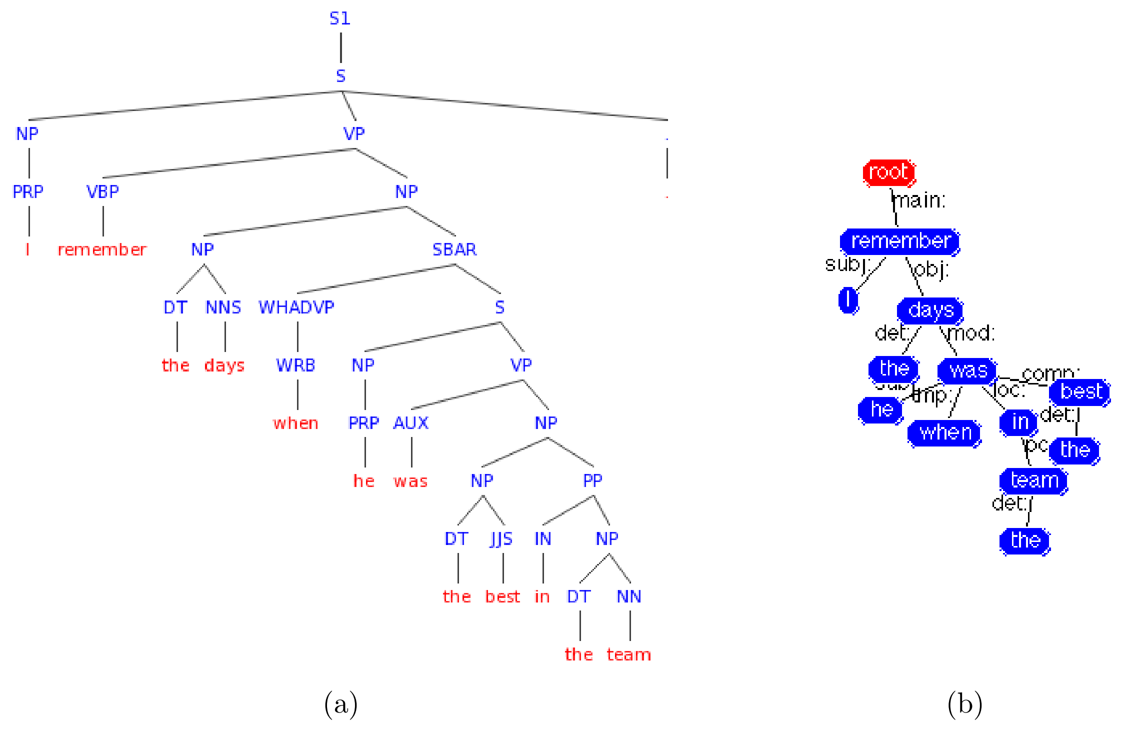


Figure 5.3: A phrase structure tree (a) and a dependency tree (b) for a sentence containing a temporal expression with a dependent clause within its extent.

examination, we determine should not be used as triggers (143 terms); the decisions made here are to some extent subjective, but are based on the heuristic that terms chosen as triggers must be clearly time-oriented (e.g. month names). Examples of words that were removed at this point are as follows: *Buddha*, *McDonald*, *nineteen*, *479* and *60th*; these appeared within temporal expressions in the following sentences and fragments:

(5.11) a. ...who were protesting against the ban on the Buddhist flag on Vesak, *the Buddha's birthday*.

b. The explosion comes *a month after a bomb exploded at a McDonald's restaurant in Istanbul*, causing damage but no injuries.

c. It is widely held that the average U.S. serviceman was *nineteen years old*.

d. in f- uh, *nineteen forty eight*, I think it was.

e. Second invasion of Greece (*480-479 BC*)

f. Dr Vittorio Sacerdoti has told his remarkable tale on *the 60th anniversary of liberation of Rome*.

In the next step we look at terms that appear both inside and outside of temporal expressions. We could discard these on the grounds that they are ambiguous, in that sometimes they are used in a non-temporal sense. However, some preliminary testing showed that doing so would remove many very valuable triggers; some words appear within temporal expressions in over 100 instances, but appear outside only once.[9] This

[9]Not infrequently this is in fact due to an omission error in the gold standard. But there are also cases where a common temporal trigger word may be used in a non-temporal way: for example, *Day* can appear as a person's last name.

```
[Day,]YYYY/MM/DD [HH:MM:SS] [TMZ]       [Day,]YYYY-MM-DD [HH:MM:SS] [TMZ]
[Day,]DD/MM/YYYY [HH:MM:SS] [TMZ]       [Day,]DD-MM-YYYY [HH:MM:SS] [TMZ]

HH:MM[:SS] [AMPM] [TMZ]
HH AMPM [TMZ]
HH o'clock [TMZ]

[Day,] YYYY Month DD [HH:MM:SS] [TMZ]  [Day,] DD Month YYYY [HH:MM:SS] [TMZ]
[Day,] DD/Month[/YYYY]                 [Day,] DD-Month[-YYYY]
[Day,] Month/DD[/YYYY]                 [Day,] Month-DD[-YYYY]
[Day,] [YYYY/]Month/DD                 [Day,] [YYYY-]Month-DD
Month/YYYY                             Month-YYYY
```

Figure 5.4: Date and time formats matched by our Trigger Tagger.

would inevitably exclude many expressions from being tested in the experiment, so we decided to manually filter these cases by picking as triggers those terms which (a) occur more often inside than outside of temporal expressions, (b) are clearly related to time, and (c) are not excluded by the TIMEX2 guidelines (e.g. *later* is related to time, but is a non-trigger).[10] So, for example, *18* and *Prophet* would not be chosen as triggers, although they appeared more often inside temporal expressions (see Example (5.12)) than outside.

(5.12) a. The Persians then counterattacked, and the Athenian force was itself besieged for *18 months*, before being wiped out.

b. Thousands of Iraq's majority Shiite Muslims marched to their main mosque in Baghdad to mark *the birthday of Islam's founder Prophet Mohammed.*

As a result, we obtained a final set of 166 word triggers, which are listed in Table 5.1.

Using the approach just described to derive a trigger list is likely to produce quite different statistics, and therefore quite different sets of triggers, depending upon the domain or corpus used. We generated our statistics on the basis of our two corpora combined into a single dataset, since we want to work with the same set of triggers for both corpora. We obtained very similar figures for coverage and precision of the triggers for both corpora; we discuss these results below.

Our analysis also showed that four digit numbers often appear within the extents, denoting both years (e.g. *1997*) and hours (e.g. *0545* refers to 5:45am). The interpretation of bare numbers is in general highly ambiguous and domain dependent, so to check the usefulness of such triggers we consider them as a separate class and also evaluate their performance separately. However, we limit the range of four digit numbers considered to the range 1900–2019; this makes our treatment consistent with Ahn et al. (2005a) in this regard.[11]

A specific class of triggers not considered in the above procedure are conventionalised patterns consisting of several tokens separated by punctuation characters: for

[10]TIMEX2 is not very precise when it comes to defining the criteria for treating time-related words as non-triggers: a brief explanation only says that they are 'less amenable to being pinned down to a timeline'. In verifying whether a term is considered 'non-markable' we used the examples of such terms found in the guidelines (Ferro et al., 2005, pp. 7–8).

[11]We note that the restriction to the 20th and the beginning of the 21th century is not well-suited to WikiWars, which also contains bare year references to earlier years.

afternoon	afternoons	ago	anniverary	anniversary	annual	Annually
annually	April	april	Aug	August	august	autumn
Autumn	birthday	centuries	century	Christmas	current	currently
Currently	daily	dawn	day	Day	days	Days
daytime	Dec	decade	decades	December	december	dusk
Easter	easter	eighties	era	Era	Eve	eve
evening	evenings	everyday	Feb	February	february	former
Former	fortnight	Fri	Friday	friday	future	Future
FUTURE	Halloween	Hanukkah	hour	hours	Hours	hr
hrs	Jan	January	january	July	july	June
june	lately	March	May	midday	midnight	millenium
millennium	minute	minutes	Minutes	Mon	Monday	monday
month	Month	monthly	Monthly	months	Months	morning
mornings	night	nights	nighttime	noon	Nov	November
november	now	NOW	Oct	October	october	overnight
Overnight	period	Period	periods	presently	quarterly	recent
Recent	recently	Recently	Sat	Saturday	saturday	semiannual
Sept	September	september	sixties	spring	Spring	summer
Summer	Sun	Sunday	sunday	Thanksgiving	Thirties	Thu
Thursday	thursday	today	Today	tomorrow	Tomorrow	tonight
Tue	Tuesday	tuesday	Wed	Wednesday	wednesday	week
Week	weekend	weekends	weeklong	weekly	weeks	winter
Winter	winters	workday	year	Year	YEAR	years
Years	yesterday	Yesterday	yesterdays	yrs		

Table 5.1: The set of triggers used in the syntactically-aware extent recognition experiment.

example, *17-07-1964* and *18:16:32*. We recognize these triggers with rules matching the patterns presented in Figure 5.4; the parts of patterns in square brackets are optional and the `TMZ` stands for a time difference (in the format of four digits after a plus or minus sign) or a time zone code (only those that we found in our corpora, i.e. *gmt*, *GMT*, *UTC*, *PTS*, *EST*). As there are dates which mix the conventionalised patterns with names of months and weekdays (e.g. *Wednesday, 17-Jan-2001*), which we already select as triggers based on the analysis of the content of the gold-standard annotations, we need to augment the patterns to avoid generating two triggers for such cases. Therefore, we allow full and abbreviated names of months and weekdays (`Month` and `Day` in the patterns, respectively) to appear within the patterns. Another reason for doing this is that some syntactic parsers treat such expressions as single syntactic nodes. The actual grammar is also slightly more complex than the one presented in Figure 5.4, as some commas, dashes and slashes in the patterns are made optional and some numbers marked here by double letters (e.g. `MM`) are permitted to be single digits as well. In consequence, the rules match expressions that could be recognized syntactically, such as *13 January 2001*; however, since one of our test parsers (Connexor) identifies such expressions as single nodes, we decided that it would be better not to use different sets of triggers and patterns for the different parsers used in the experiment.

					Strict			Lenient		
Set	Corr.	Miss.	Spur.	Part.	Prec.	Recall	F	Prec.	Recall	F
Words	1567	1563	641	2298	0.35	0.29	0.32	0.86	0.71	0.78
Dates	875	4327	67	226	0.75	0.16	0.27	0.94	0.20	0.33
Years	198	4307	46	923	0.17	0.04	0.06	0.96	0.21	0.34
Words+Dates	2442	849	534	2137	0.48	0.45	0.46	0.90	0.84	0.87
Words+Years	1765	755	1000	2908	0.31	0.33	0.32	0.82	0.86	0.84
Dates+Years	1073	4062	111	293	0.73	0.20	0.31	0.92	0.25	0.40
Words+Dates+Years	2640	605	599	2183	0.49	0.49	0.49	0.89	0.89	0.89

Table 5.2: The results obtained on the ACE corpus with the Trigger Tagger.

					Strict			Lenient		
Set	Corr.	Miss.	Spur.	Part.	Prec.	Recall	F	Prec.	Recall	F
Words	348	755	61	1578	0.18	0.13	0.15	0.97	0.72	0.83
Dates	772	1857	0	52	0.94	0.29	0.44	1.00	0.31	0.47
Years	319	1744	0	618	0.34	0.12	0.18	1.00	0.35	0.52
Words+Dates	1120	722	62	839	0.55	0.42	0.48	0.97	0.73	0.83
Words+Years	667	337	580	1677	0.23	0.25	0.24	0.80	0.87	0.84
Dates+Years	1091	1334	5	256	0.81	0.41	0.54	1.00	0.50	0.67
Words+Dates+Years	1439	326	194	916	0.56	0.54	0.55	0.92	0.88	0.90

Table 5.3: The results obtained on the WikiWars corpus with the Trigger Tagger.

Evaluation of the derived set of triggers Once we have chosen the set of triggers, we check how well these perform for the recognition of temporal expressions without growing the extents. These results determine the lower and upper bounds for the approaches based on syntactic analysis. The strict results for recognizing triggers only provide a lower bound, allowing us to see how much of improvement we can obtain by using a parser; the lenient results, on the other hand, are the upper limit of what we can expect to achieve in the experiment (i.e. they provide the score we would achieve if every detected expression received a correct extent).

In Table 5.2 and Table 5.3 we present the lenient and strict results obtained with a tagger which recognizes only the triggers—which we refer to as the Trigger Tagger—for the ACE 2005 Training and WikiWars corpora, respectively. We consider the three classes of triggers identified above individually and together. Apart from precision, recall and F-measure, we also report the number of correctly recognized expressions (the 'Corr.' column), the number of expressions recognized only partially (the 'Part.' column), and also the number of missed and spuriously annotated (false positive) expressions (the 'Miss.' and 'Spur.' columns, respectively).

As anticipated, we get the best performance with all the triggers used together ('Words+Dates+Years'); for both corpora we get very good coverage in detection (0.89 and 0.88 lenient recall) without compromising the precision too much (0.89 and 0.92) which results in high lenient F-measures: 0.89 and 0.90. It is encouraging that, with the same set of triggers, we obtained such similar results for detection on both, quite different, corpora; there was a risk that by removing some words because of their statistics on one of the datasets we might lower the performance on the other dataset for which they might be good triggers. We also note that on both corpora all the three sets of triggers deliver mutually exclusive sets of correctly recognized expressions: the number for 'Words+Dates+Years' configuration is the exact sum of the 'Corr.' measures obtained for individual subsets.

In the case of word triggers we notice that the total number of spurious matches (641+61=702) is much higher than the total number of occurrences of these words outside of temporal expressions (235). This is because, apart from annotating words from outside of temporal expressions, we also sometimes get two (or possibly even more) matches within a single expression. For example, *the night of 8 February 1904* contains two words that we have in our set of triggers: *night* and *February*.

The relatively high strict recall (0.49 and 0.54) tells us that about half of the temporal expressions appearing in texts (in our corpora) do not have very complex structure; they can be correctly recognized just with word triggers and a quite limited set of patterns. This should be kept in mind when evaluating any of the fuller-featured taggers. Matching just trigger words yields 0.29 and 0.13 strict recall on our data; the high difference shows that differences among datasets may be significant.

5.4 The Dependency-based Approach

We first experiment with the approach based on functional dependencies. This is more straightforward to implement, on the assumption that the trigger is the head of the expression; without such an assumption, or when using phrase structures, we need to choose one from a number of possible candidates (see, for example, the tree in Figure 5.3a) whereas in this model we have immediately only one candidate (see the corresponding tree in Figure 5.3b).

The overall structure of the algorithm used in our method is presented in Figure 5.5. We take the trigger word, and by recursively following the links to the children until we reach the leaves of the dependency tree we collect all the tokens in a list. When the whole tree is traversed we sort the list by the location (character offset) of each token in the sentence. Then, we take the first and last tokens from the list and make the annotation on the span of text denoted by the two tokens.

By making the annotation between the two border tokens, instead of explicitly including all the tokens from the subtree one by one, we ensure that we have a contiguous sequence of tokens, which is required by the definition of temporal expression that we have adopted. This means that we can recover from some potential parsing errors, in those cases where the parser does not find the dependency relations to one or more tokens inside the expression. For example, although the analysis of the sentence from Example (5.13) obtained from Connexor does not include *especially in places* in the subtree of the trigger (*time*), the correct extent is found because the last token *Diem* is included in the subtree.

(5.13) The pledge came at *a* ***time*** *when Vietnam was deteriorating, especially in places like the Mekong Delta, because of the recent coup against Diem.*

Unfortunately, we will not recover from the opposite situation, when a parser generates spurious dependencies to tokens not belonging to the expression; in such cases the method will generate an incorrect extent. We also assume that, when fully integrated into a processing pipeline, the trigger on the input would already be disambiguated with regard to whether it really denotes a temporal expression or not (e.g. *The Long*

```
FIND_EXTENT($trigger$)
 1  $toProcess \leftarrow \{\}$; $extentNodes \leftarrow \{\}$
 2  PUSH($toProcess$, $trigger$)
 3  INSERT($extentNodes$, $trigger$)
 4  repeat
 5          $nextNode \leftarrow$ POP($toProcess$)
 6          PUSH($toProcess$, CHILDREN($nextNode$))
 7          INSERT($extentNodes$, CHILDREN($nextNode$))
 8    until IS_EMPTY($toProcess$)
 9  SORT($extentNodes$)
10  MAKE_ANNOTATION(FIRST($extentNodes$), LAST($extentNodes$))
```

Figure 5.5: The algorithm for finding the extent of a temporal expression headed by a trigger using a dependency tree.

March), and so the algorithm does not attempt to check this; this contributes to the false positives in the numbers reported here.

5.4.1 The Parsers

In our experiments we use four off-the-shelf parsers: Minipar,[12] Connexor,[13] Stanford parser,[14] and C&C.[15] Connexor is a commercial tool for which we obtained a license to use in this research; all the other parsers are developed by academic research laboratories and are publicly available on the web for download. We do not retrain any of the parsers because we do not have enough training data; this is also a common scenario in real-life applications when one has no choice but to use an off-the-shelf parser. Our experiments are carried out using the GATE framework, which already includes wrappers for the Minipar and Stanford parsers; to integrate Connexor and C&C within our framework we implemented our own wrappers.

The chosen parsers have different requirements for input data. Connexor carries out its own sentence splitting (see Table 5.4 for how this impacts the number of sentences identified in the corpora used in the experiments), while the other parsers expect the text to be already segmented into sentences. Another area of difference is tokenisation; only the C&C parser expects the text to be already tokenized,[16] while all the other parsers do their own tokenization. Furthermore, Connexor can combine several words into a single token (and, in consequence, syntactic node), e.g. *Cape Town*, *a.m.*; this goes as far as treating even expressions like *25 December 1956* as a single node.

In the initial development phase we discovered some issues with some parsers, and made relevant adjustments. For example, Minipar treats commas as syntactic nodes being in functional relations with other tokens. This can lead to the incorrect inclusion of a comma in the extent of a temporal expression as its last token, for example after *June* and *August* in Example (5.14).

[12]See `http://webdocs.cs.ualberta.ca/~lindek/minipar.htm`.

[13]See `http://www.connexor.eu/technology/machinese/machinesesyntax`.

[14]We used version 1.6.5; see `http://nlp.stanford.edu/software/lex-parser.shtml`.

[15]See `http://svn.ask.it.usyd.edu.au/trac/candc`.

[16]The expected style is that of the Penn Treebank, so we used the tokenizer found at the Penn Treebank website (see `http://www.cis.upenn.edu/~treebank/tokenization.html`); however, we introduced some minor modifications (e.g. to avoid breaking numbers like *3,000* into separate tokens).

	ACE 2005 Training	WikiWars
Documents	593	22
Tokens (Alternate Tokeniser)	322k	121k
Sentences (GATE)	18,252	4,869
Sentences (Connexor)	18,843	4,857
TIMEX2	5,428	2,681

Table 5.4: A comparison of the size of the datasets used in the experiments on extent recognition with the use of syntactic parsers.

(5.14) ...initiating a siege of Malta in *June*, conquering British Somaliland in *August*, and making ...

We fixed this issue by implementing an additional postprocessing step that shortens the annotation by one character when the last token of the extent of a temporal expression is a comma.[17]

As mentioned earlier, the Stanford parser uses a very rich set of 48 dependency categories. This in consequence introduces dependencies which we would not expect to find. Based on an analysis of the categories of the Stanford Dependencies (described in the manual by de Marneffe and Manning (2008)) and a few example parses of sentences from our data we decided not to follow nsubj (nominal subject), cop (copula), cc (coordination), and conj (conjuct); otherwise we would not be processing dependency trees, but dependency graphs.

5.4.2 Results

5.4.2.1 Applying the Method to Individual Subsets of Triggers

We first applied the method of using dependency trees for extent recognition to each of the three subsets of triggers individually; for the two corpora we use, these results are shown as the first three set-ups in Table 5.5 and Table 5.6. As expected, the lenient results have not really changed compared to using just triggers (see Table 5.2 and Table 5.3), as these reflect only the performance for the detection task;[18] and, consequently, of course, the choice of the parser did not matter here.

With regard to recognition, the strict results naturally differ across the parsers; however, the differences in many cases are insignificant. Despite the differences, we can see that with the use of the syntax-aware method the strict F-measure got (a) much higher when using word-triggers, (b) much lower when using conventionalised-pattern-based dates as triggers, and (c) slightly worse or almost unchanged (except for the set-up with Minipar on WikiWars, where it increased a lot) for four digit numbers (years).

[17]Our post-experimental investigation revealed that this improved the results on WikiWars, but had almost no impact on the ACE corpus.

[18]There are small changes in the number of missing and spurious expressions but in most cases they are not big enough to impact the lenient precision, recall and F-measure: only in two configurations (set-up/parser) for WikiWars the difference was 0.1. These changes occur because after growing the extent with the syntactic method, the matchings between system-generated annotations and those found in the gold standard changed slightly compared to when we used triggers without the application of the method.

Set-up		Measure	Minipar*	Connexor	Stanford	C&C
1	(Words)	Correct	2050	2373	2292	2505
		Missing	1562	1559	1553	1554
		Spurious	640	637	631	632
		Part. Correct	1816	1496	1583	1369
		Strict P/R/F	0.45/0.38/0.41	0.53/0.44/0.48	0.51/0.42/0.46	0.56/0.46/0.50
		Lenient P/R/F	0.86/0.71/0.78	0.86/0.71/0.78	0.86/0.71/0.78	0.86/0.71/0.78
2	(Dates)	Correct	586	630	623	390
		Missing	4330	4327	4327	4330
		Spurious	67	67	67	67
		Part. Correct	512	471	478	708
		Strict P/R/F	0.50/0.11/0.18	0.54/0.12/0.19	0.53/0.11/0.19	0.33/0.07/0.12
		Lenient P/R/F	0.94/0.20/0.33	0.94/0.20/0.33	0.94/0.20/0.33	0.94/0.20/0.33
3	(Years)	Correct	249	208	196	273
		Missing	4307	4306	4307	4308
		Spurious	43	45	46	44
		Part. Correct	872	914	925	847
		Strict P/R/F	0.21/0.05/0.08	0.18/0.04/0.06	0.17/0.04/0.06	0.23/0.05/0.08
		Lenient P/R/F	0.96/0.21/0.34	0.96/0.21/0.34	0.96/0.21/0.34	0.96/0.21/0.34
4	(Words+Year) +Dates	Correct	3124	3377	3301	3514
		Missing	605	603	605	605
		Spurious	596	597	599	596
		Part. Correct	1699	1448	1522	1309
		Strict P/R/F	0.58/0.58/0.58	0.62/0.62/0.62	0.61/0.61/0.61	0.65/0.65/0.65
		Lenient P/R/F	0.89/0.89/0.89	0.89/0.89/0.89	0.89/0.89/0.89	0.89/0.89/0.89
5	(Words) +Year+Dates	Correct	3121	3388	3304	3511
		Missing	604	603	605	604
		Spurious	598	597	599	598
		Part. Correct	1703	1437	1519	1313
		Strict P/R/F	0.58/0.57/0.58	0.62/0.62/0.62	0.61/0.61/0.61	0.65/0.65/0.65
		Lenient P/R/F	0.89/0.89/0.89	0.89/0.89/0.89	0.89/0.89/0.89	0.89/0.89/0.89

Table 5.5: Results for extent recognition obtained with the dependency-based approach on the ACE 2005 Training corpus.

This suggests that word triggers (e.g. *month* and *Monday*) are good candidates for heads of temporal expressions, but pattern-based dates and times (e.g. *17/01/2001* or *7:05 a.m.*) are not and, as expected, they seem to be not extendable (i.e. they already constitute 'saturated' expressions). The error analysis also showed that dates often appear in date ranges (e.g. *5 September-12 September*) which are incorrectly tokenised by the parsers; this problem does not occur if the dates are annotated as temporal expressions already by the conventionalised-pattern recognition rules. In the case of years denoted by four-digit numbers we obtained a big improvement with Minipar on WikiWars because in this case the method correctly extended the annotations of expressions consisting of a month name and a year number (e.g. *April 1944*); these are common in this dataset. The other parsers considered the month name, rather than the year, to be the head in such phrases. Overall, performance with using these parsers decreased because, in addition to not positing the years as the heads of the month names, they made a number of errors in recognizing dependencies in sentences, resulting in occasional inclusion in the extents of additional tokens from the vicinity of the standalone years. Without these deficiencies in the parsers the years would also be good triggers in other cases where they are the syntactic heads of phrases, e.g. *early*

Set-up		Measure		Minipar*	Connexor	Stanford	C&C
1	(Words)	Correct		705	1365	1189	1369
		Missing		755	754	753	753
		Spurious		61	60	59	59
		Part. Correct		1221	562	739	559
		Strict	P/R/F	0.35/0.26/0.30	0.69/0.51/0.58	0.60/0.44/0.51	0.69/0.51/0.59
		Lenient	P/R/F	0.97/0.72/0.83	0.97/0.72/0.83	0.97/0.72/0.83	0.97/0.72/0.83
2	(Dates)	Correct		697	700	718	674
		Missing		1857	1857	1857	1857
		Spurious		0	0	0	0
		Part. Correct		127	124	106	150
		Strict	P/R/F	0.85/0.26/0.40	0.85/0.26/0.40	0.87/0.27/0.41	0.82/0.25/0.38
		Lenient	P/R/F	1.00/0.31/0.47	1.00/0.31/0.47	1.00/0.31/0.47	1.00/0.31/0.47
3	(Years)	Correct		626	312	306	248
		Missing		1773	1744	1744	1773
		Spurious		0	0	0	0
		Part. Correct		282	625	631	660
		Strict	P/R/F	0.69/0.23/0.35	0.33/0.12/0.17	0.33/0.11/0.17	0.27/0.09/0.14
		Lenient	P/R/F	1.00/0.34/0.51	1.00/0.35/0.52	1.00/0.35/0.52	1.00/0.34/0.51
4	(Words+Year) +Dates	Correct		1708	1775	1850	1836
		Missing		355	325	326	354
		Spurious		194	193	194	193
		Part. Correct		618	581	505	491
		Strict	P/R/F	0.68/0.64/0.66	0.70/0.66/0.68	0.73/0.69/0.71	0.73/0.68/0.71
		Lenient	P/R/F	0.92/0.87/0.89	0.92/0.88/0.90	0.92/0.88/0.90	0.92/0.87/0.89
5	(Words) +Year+Dates	Correct		1677	1818	1864	1910
		Missing		326	325	326	325
		Spurious		194	193	194	193
		Part. Correct		678	538	491	446
		Strict	P/R/F	0.66/0.63/0.64	0.71/0.68/0.70	0.73/0.70/0.71	0.75/0.71/0.73
		Lenient	P/R/F	0.92/0.88/0.90	0.92/0.88/0.90	0.92/0.88/0.90	0.92/0.88/0.90

Table 5.6: Results for extent recognition obtained with the dependency-based approach on WikiWars.

1996.

It is interesting to note that trigger words on their own yield better strict results on ACE than WikiWars (F-measure: 0.32 vs 0.15), but after applying the syntactic method, the situation changes: the performance is higher on WikiWars with a gain in F-measure of 0.44, compared to only 0.18 on ACE. This means that in ACE there are (proportionally) more expressions which are built just with trigger words, but in WikiWars either there are (proportionally) more multi-word expressions headed by word triggers, or the parsers perform better, which might in turn mean that the sentences are more grammatical and easier to parse.[19]

[19]The ACE corpus contains a number of documents with transcribed speech, blog entries and UseNet discussions. These text genres are challenging to parse; the first is likely to have errors introduced by the speech-to-text technology, and the other two are usually not as carefully edited texts as those professionally prepared for publication (e.g. news, books). WikiWars, on the other hand, is sourced from Wikipedia, which has some quality verification mechanisms implemented to eliminate errors.

5.4.2.2 Applying the Method to Combined Subsets of Triggers

Based on the initial results described above we decided to run two set-ups (which we refer to here as #4 and #5) combining all three sets of triggers. In both set-ups the words are used as the syntactic heads of the expressions, and the date-recognition rules produce final annotations of temporal expressions (i.e. they are not subject to the syntax-driven method). What distinguishes the set-ups is the use of four digit numbers; in the first set-up we attempt to grow the extents around these, in the other set-up we treat them as complete temporal expressions.

The results for these set-ups are also shown in Table 5.5 and Table 5.6. First of all we cannot tell from the results whether it is better to use the four-digit numbers as triggers or complete expressions; however, when using the numbers as triggers (set-up #4) on WikiWars the lenient results dropped by 0.01, caused by an increase in the missing measure of 28 (for C&C) and 29 (for Minipar); otherwise, they stayed unchanged. On both corpora the best strict results were obtained with the C&C parser, and the worst with Minipar. The taggers using Stanford and Connexor are placed in the middle, but the differences here are not big, and depending on the corpus, one or the other parser seems to be more useful.

As the best results were obtained with the C&C parser, most of our analysis of the results will concern a tagger based on this parser and set-up #5. As with running the method for individual subsets of triggers, the absolute results for WikiWars were higher than for ACE: on WikiWars the tagger scored 0.75 strict precision, 0.71 recall and 0.73 F-measure; on ACE, both precision and recall were 0.65, thus F-measure was the same too. The precision being the same as recall means that the system generated the same number of annotations, or very close to, as there are in the gold standard.[20] However, we are interested not only in the absolute values, but also in the gain obtained with the syntactic method compared to recognizing triggers alone, since this shows how much value the dependency trees add in extent recognition.

5.4.2.3 The Gains from the Method

If we compare the results in set-up no. 1 with just annotating the word triggers as temporal expressions, the gains are quite different for the two corpora: 0.38 for WikiWars and 0.17 for ACE (recall), and 0.44 for WikiWars and 0.18 for ACE (F-measure); see the first two columns in Figure 5.6 (the gains obtained from applying the syntactic information are marked with a bold frame). The gain on ACE is smaller because, as we noted before, there are more expressions whose extent consists only of the trigger, and in many documents the sentences are more difficult to parse. The gain obtained on WikiWars is satisfactorily high and shows that the method can successfully find the correct extent in many cases. However, while the lenient recall was 0.72, the strict corresponding measure was 0.51; this means that for about 29% of the detected expressions, the method did not find the exact extent.

If we compare the results from set-up no. 5 with those obtained when annotating the three sets of triggers as temporal expressions, the gains for WikiWars are much smaller than they were for set-up no. 1, and are now very similar for the two corpora: 0.17 for WikiWars and 0.16 for ACE (recall), and 0.18 for WikiWars and 0.16 for ACE

[20]This follows straightforwardly from the definitions of precision and recall: $\mathrm{P} = \frac{\#\text{correct}}{\#\text{generated}}$ and $\mathrm{R} = \frac{\#\text{correct}}{\#\text{gold-standard}}$.

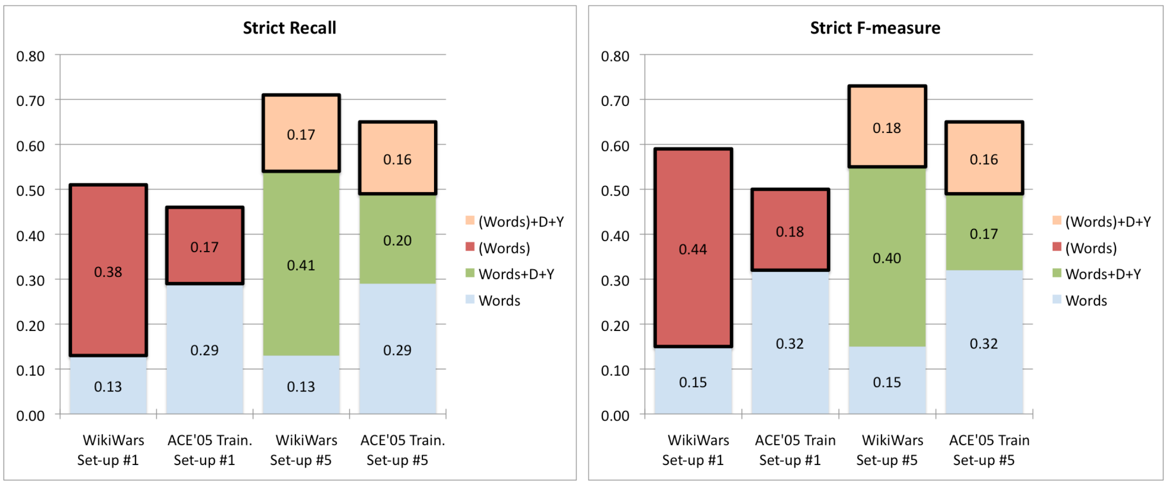


Figure 5.6: The gains in results scored by the dependency-based approach on the ACE 2005 Training and WikiWars corpora.

(F-measure). The gain for WikiWars is now much smaller because many temporal expressions are now matched by the rules annotating dates and times in conventionalised formats.

5.4.2.4 Comparison to Pattern-based Taggers

Although the results just discussed prove the method to be working as anticipated, we note that there is still a big gap between the strict and lenient results: 0.24 on ACE and 0.17 on WikiWars (F-measure). This means that in a number of cases the method did not produce the correct extent. Furthermore, the absolute performance is lower than what we can obtain with a pattern-based tagger; in Table 5.7 we compare the best result of the syntax-driven method (i.e. set-up #5 with the C&C parser) with the DANTE (Mazur and Dale, 2007) tagger, presented in Chapter 7. In Table 5.8 and Table 5.9 we provide the confusion matrices for the dependency-based approach based on the C&C parser and DANTE, both run on WikiWars; Table 5.8 shows the results for set-up no. 1 and Table 5.9 for set-up no. 5. Expressions that were recognized by both systems are the most common category.

From the point of view of determining the added value that the dependency-based approach provides, the cases of particular importance for us are those where DANTE failed to recognize the extent of the expression correctly, but the dependency-based approach succeeded: depending on the set-up there are either 81 or 83 such cases which constitutes about 55% of all incorrectly recognized extents by DANTE. Most (68) of these expressions are event-based (e.g. *the first hour of the attack*); these are indeed very challenging for pattern-based approaches. Six other expressions included hyphenated tokens (e.g. *month-long*, *present-day*, *odd-numbered days*), which might have posed a problem with tokenisation for the recognition rules of DANTE; the remaining seven expressions were *only one week*, *future*, *the day following*, *several days following*, *two bloody weeks*, *only a few months before* and *only a few days earlier*.

We also note the high number of expressions correctly recognized by DANTE but incorrectly recognized by the dependency-based approach: 501 for set-up no. 1 and

						Strict			Lenient		
		Corr.	Miss.	Spur.	Part.	Prec.	Recall	F	Prec.	Recall	F
ACE05Train	DepC&C	3511	604	598	1313	0.65	0.65	0.65	0.89	0.89	0.89
	DANTE	4269	417	709	742	0.75	0.79	0.77	0.88	0.92	0.90
WikiWars	DepC&C	1910	325	193	446	0.75	0.71	0.73	0.92	0.88	0.90
	DANTE	2494	36	44	151	0.93	0.93	0.93	0.98	0.99	0.99

Table 5.7: A comparison of recognition results for the dependency-based approach and a pattern-based tagger.

		DANTE		
		Corr.	Part.	Miss.
C&C	Corr.	1273	81	16
	Part.	501	52	5
	Miss.	720	18	15

Table 5.8: The confusion matrix for the output of DANTE and the dependency-based approach with the C&C parser run on the WikiWars corpus (set-up #1).

		DANTE		
		Corr.	Part.	Miss.
C&C	Corr.	1813	83	15
	Part.	389	51	5
	Miss.	292	17	16

Table 5.9: The confusion matrix for the output of DANTE and the dependency-based approach with the C&C parser run on the WikiWars corpus (set-up #5).

389 for set-up no. 5; this equals to about 90% of the errors made by the dependency-based approach in the particular set-up. Analysing the results for set-up no. 1, we notice that the majority of expressions concern dates in various formats, for example *24 September*, *June 4*, *January 1955*, *May 8, 1945*, *August 480 BC*. This may suggest that the dependency-based approach is not particularly well suited to recognition of temporal expressions in conventionalised formats, which, on the other hand, are relatively easy to describe by means of patterns. Many modified expressions, e.g. *early February*, are also misrecognized; this includes more syntactically complex expressions like *the middle of December*. Finally, some single-word expressions (*February*, *October*, *summer*) are among those whose extent is not determined correctly, which means the dependency-based approach includes some additional tokens that are not part of the correct extent, while recognition via patterns could not be any simpler. In the next subsection we will discuss the reasons underlying these cases of incorrect recognition.

5.4.3 Error Analysis

An analysis of the dependency-based tagger's output reveals a number of sources of error: problems with sentence splitting and tokenisation, parsing errors, the identification of heads in dependency relations, the recognition of triggers, and the choice of which token is to be considered the head of a temporal expression.

5.4.3.1 Sentence Splitting and Tokenisation

Inevitably, in a large corpus containing documents from various domains, including blogs, discussion forums and speech transcripts, some problems arise concerning sentence splitting. Any such error is very likely to negatively impact the parsing process

Wednesday, July 6th
– Leave from Niagara Falls and drive to Toronto, On – 85 miles
– Drive back to Niagara Falls *that evening* – 85 miles
– Stay in Niagara Falls

Figure 5.7: A fragment of a UseNet document which was not correctly split into sentences, thus affecting the recognition of temporal expressions; the incorrectly found extent is underlined.

---

and result in an incorrect syntactic analysis of the affected sentences. In Figure 5.7 we present an example of such a problematic text fragment: this was segmented as a single sentence, rather than being broken into one sentence per line. Consequently, the recognized extent of the expression *that evening* included a number of tokens from both its preceding and following line; in Figure 5.7 we have marked the extent found with underlining. Note that pattern-based taggers which do not make use of syntactic parsing most likely would not suffer from this problem, as they usually do not need the sentence boundaries identified and search for their patterns more locally.

The tokenisation carried out by the parser sometimes resulted in combining two or more tokens into a single syntactic node. This happened, for example, when date ranges such as *12-15 September* did not have space characters around the dash. This then resulted in too many tokens being incorporated into the extent of the temporal expression.

5.4.3.2 Parsing Errors

Of course, none of the parsers is perfect. All of them make mistakes by either producing incorrect dependencies or missing some. Our extent recognition process is very sensitive to such errors; a single spurious dependency whose head is a token from within the correct extent of the expression, but whose dependent is outside the correct extent already results in an incorrect annotation.

One of the problems is the attachment of prepositional phrases and dependent clauses, as shown in Example (5.15); here we have marked the correct extents with italics and the extents found by the tagger using underlining. For instance, in Example (5.15a), the tagger annotated only *the bloodiest day* because *in its history* was attached by the parser to the verb *endured* instead of to the noun phrase *the bloodiest day*.

(5.15) a. The British Army endured *the bloodiest day in its history*, suffering 57,470 casualties…

b. …surrendering *only two weeks after the armistice took effect in Europe.*

c. In *the autumn of 1917*, thanks to the improving situation on the Eastern front, the Austro-Hungarian troops…

In other cases the parses are broken and the extents of the temporal expressions include extra tokens from a phrase following the expression itself:

(5.16) a. …after stalling for *a day* against the main resistance line to which the enemy had withdrawn.

b. …believing it would be *many more months before they would arrive* and that the arrival could be stopped by U-boats.

c. . . . a decisive blow for *the next two years*, though protracted German action at Verdun throughout 1916,. . .

d. . . . the U.S. Congress passed the Crittenden-Johnson Resolution on *July 25 of that year*, which stated that the war was being fought to preserve the Union and not to end slavery.

Of course, it is possible that if we had retrained the parser on more similar data these problems might be reduced.

5.4.3.3 Heads in Dependencies

Another issue related to parsing is the determination of what the head is in a dependency structure. For example, in Connexor's analysis of *more than 20 years of war*, *more* is considered to be the head of the tree, rather than the more intuitive choice of *year*. Determining the specific role of a token in a dependency relation is a more general problem where a consensus is not alway available: see, for example, (Xia and Palmer, 2001, pp. 1–2).

Some errors which occur when using the C&C parser are related to the production of dependencies when one node can have two different heads; this occurs, for example, in relation to conjunctions. For instance, C&C's analysis of *the spring and early summer of 1943* makes each of the tokens *the* and *of* children in two dependencies: det(spring,the), det(summer,the), det(spring,of), and det(summer,of). This means that the output in such cases is a directed graph, not a tree. In consequence, for both triggers, *spring* and *summer*, the annotations we find span over the whole string *the spring and early summer of 1943.*

5.4.3.4 Triggers

Recognizing dates and years as saturated expressions (i.e. not applying the dependency-based approach to further extend them) also contributed to a number of errors, e.g. in expressions referring to decades (*the 1950s*) or modified dates (*early September 1950*). However, as we discussed earlier, extending these triggers did not improve the overall results: while in such a set-up some expressions are provided with the correct extents, others include tokens from outside of the correct extent because of parsing errors.

We also found that, while the set of word triggers did not include *March* and *May* because of their ambiguity, we accidentally also removed these two month names in the date recognition patterns, where the surrounding tokens would have provided sufficient disambiguation. This omission resulted in 53 cases where the dependency-based approach was applied instead of the date patterns, resulting in incorrect extents either because of parsing errors or the treatment of heads in dependencies.

5.4.3.5 Heads of Temporal Expressions

Finally, we also found that the method itself can be a source of errors. We noticed that, independently of the particular parser used, for many expressions the extent did not include tokens preceding the trigger. For example, for the expression *the middle of August* only *August* was annotated. It turns out that the TIMEX2 stipulation that the trigger is the syntactic head does not hold for all cases;[21] for the given example,

[21] Alternatively, one might take the view that this is simply a disagreement between TIMEX2 and the parsers as to what the head of the expression is.

the dependencies are middle:>the, middle:>of and of:>August, so that *middle* is the head of the expression as a whole and the trigger (*August*) is a leaf in the tree. Our analysis of the results showed that in WikiWars there are 64 expressions based on the pattern 'the [start|middle|end|...] of <trigger>', of which 26 use a date or a year as a trigger; this constitutes 14% of all incorrectly recognized expressions. Given the relatively constrained structure of such cases, the issue could be potentially fixed by checking the ancestors of the trigger and adding the missing tokens.[22]

Another similar issue concerns expressions like *the 28th of that month*, where the head is *28th* rather than *month*; since we did not use the ordinal numbers as triggers in our experiments, all expressions with this structure were recognized only partially.

5.4.4 Extent Recognition of Event-based Expressions

Extent recognition in the case of event-based expressions is more challenging than for other types of expressions because their extent usually contains dependent clauses of arbitrary complexity and length. It seems practically impossible to write a lexicalized pattern-based recognition grammar for these expressions, because the extent may contain any words from the language. Consider the following examples from WikiWars:

(5.17) a. the day after the US's northward crossing of the 38th-parallel border into North Korea

b. a time when Vietnam was deteriorating, especially in places like the Mekong Delta, because of the recent coup against Diem

A finite-state pattern matcher does not seem to be the proper way to address the problem. It seems that the only reasonable approach is to use a syntactic parser to find the extent of these temporal expressions.

To further test our dependency-based approach on data of this kind we created two additional datasets. The first, which we call TG-Events, contains 25 event-based expressions found in Section 4.8 of the TIMEX2 guidelines, where this type of temporal expressions is discussed. The second dataset, WW-Events, is sourced from WikiWars and contains 135 expressions, 122 of which are event-based; the original sentences also contained a number of non-event-based expressions, many of which we removed except for 13 cases which we left because they are nested within event-based expressions as in the following example:

(5.18) By [the time Arnold reached Quebec City in [early November]], he had but 600 of his original 1,100 men.

First, we recognize just word triggers from the previous experiments to check their performance. Because it turns out we miss a proportionally large number of expressions (six on TG-Events and 24 on WW-Events), we add three more triggers: *time*, *duration* and *season*. With these extensions we detect all event-based expressions in both datasets, and the only expressions we miss are year numbers (e.g. *1910*) and *twelve* as part of the range expression *twelve to eighteen months*.[23] The results are presented in Table 5.10 and Table 5.11; the results obtained with the extended set of

[22]One could also attempt to recognize these expressions with a pattern-matching approach; however, this seems to be a step backwards, since we introduced the parser-based methods in order to avoid having to write recognition rules.

[23]The string *twelve to eighteen months* in TIMEX2 receives two annotations: one for *twelve* and one for *eighteen months*.

					Strict			Lenient		
Method	Corr.	Miss.	Spur.	Part.	Prec.	Recall	F	Prec.	Recall	F
Word-triggers	0	6	0	19	0.00	0.00	0.00	1.00	0.76	0.86
Word-triggers*	0	0	0	25	0.00	0.00	0.00	1.00	1.00	1.00
Word-triggers/C&C	16	6	0	3	0.84	0.64	0.73	1.00	0.76	0.86
Word-triggers*/C&C	21	0	0	4	0.84	0.84	0.84	1.00	1.00	1.00
DANTE	2	0	1	23	0.08	0.08	0.08	0.96	1.00	0.98
TERSEO	1	3	0	21	0.05	0.04	0.04	1.00	0.88	0.94

Table 5.10: The results for recognition of event-based expressions sourced from the TIMEX2 guidelines.

triggers are marked with an asterisk. The strict F-measures are 0.00 on TG-Events and 0.01 for WW-Events.[24] The two correctly recognized expressions on WW-Events are not event-based. In Table 5.12 we present the results calculated only for the event-based expressions from the WW-Events dataset; now we see that in both datasets there are obviously no event-based expressions consisting only of a trigger. Given the small number of missing and spurious annotations, we obtain lenient precision, recall and F-measure equal to 1.00 or very close it. The single spurious expression in WW-Events is the Chinese person name *Sun*.

When applying the syntactic method to the word triggers we experiment with the same four parsers as before. Again we obtained the best recognition results with the C&C parser, and so we present here only the results obtained with this parser. On the expressions from the TIMEX2 guidelines we scored 0.84 F-measure. On WW-Events we obtained 0.67 when we consider all the temporal expressions and 0.69 when we evaluate event-based expressions only. Although we cannot compare these values to those obtained for the full ACE and WikiWars corpora, we note that the absolute values are not higher. More important is the difference between the results of the triggers-only approach and those obtained by using the dependency-based approach: As we now process the syntactically complex expressions, the gain is now much greater than in the previous experiments.

As we cannot expect the triggers on their own to recognize any of the event-based expressions, in order to make the evaluation more meaningful, we compare the results obtained with the dependency-based approach to those of the DANTE system. As we learnt from the previous experiments, DANTE has quite decent recognition performance and for the whole ACE and WikiWars corpora it performed significantly better than the dependency-based approach. Here, DANTE scored only 0.08 in strict F-measure on TG-Events and 0.06 on event-based expressions from WW-Events. This clearly shows what we hypothesized at the beginning: a pattern-based approach is not suited to the recognition of event-based expressions.

To verify this further, we also used TERSEO (Saquete, 2005), another mature pattern-based system, and its performance on these datasets turned out to be very close to that of DANTE.[25] From the event-based expressions, TERSEO recognized only three temporal expressions: *a week of intense fighting*, *the day of national inde-*

[24]On WW-Events we observe that for evaluation with strict measures, the metrics favour missing expressions over those that are partially recognized.

[25]We used the web demo of the tagger available at `http://gplsi.dlsi.ua.es/~stela/TERSEO`.

					Strict			Lenient		
Method	Corr.	Miss.	Spur.	Part.	Prec.	Recall	F	Prec.	Recall	F
Word-triggers	2	24	1	109	0.02	0.01	0.02	0.99	0.82	0.90
Word-triggers*	2	4	1	129	0.02	0.01	0.01	0.99	0.97	0.98
Word-triggers/C&C	78	23	0	34	0.70	0.58	0.63	1.00	0.83	0.91
Word-triggers*/C&C	90	3	0	42	0.68	0.67	0.67	1.00	0.98	0.99
DANTE	20	11	2	104	0.16	0.15	0.15	0.98	0.92	0.95
TERSEO	6	23	2	106	0.05	0.04	0.05	0.98	0.83	0.90

Table 5.11: The results for extent recognition on the whole WW-Events dataset.

					Strict			Lenient		
Method	Corr.	Miss.	Spur.	Part.	Prec.	Recall	F	Prec.	Recall	F
Word-triggers	0	20	1	102	0.00	0.00	0.00	0.99	0.84	0.91
Word-triggers*	0	0	1	122	0.00	0.00	0.00	0.99	1.00	1.00
Word-triggers/C&C	72	21	1	29	0.71	0.59	0.64	0.99	0.83	0.90
Word-triggers*/C&C	84	1	1	37	0.69	0.69	0.69	0.99	0.99	0.99
DANTE	7	11	2	104	0.06	0.06	0.06	0.98	0.91	0.94
TERSEO	3	17	2	102	0.03	0.02	0.03	0.98	0.86	0.92

Table 5.12: The results for extent recognition for event-based expressions in WW-Events.

pendence, and *the time of the Revolution*. These have very simple structure which can be represented with a single pattern: Det Trigger of (JJ | Det) NN. DANTE recognized seven expressions, but none of those recognized by TERSEO: *the ceasefire period*, *the monsoon season*, *the duration of the war* (three occurrences), *the Buddha's birthday*, and *a prolonged period of appeasement*. These expressions are also quite simple syntactically. The dependency-based approach, on the other hand, successfully recognized the extents of the expressions in the following sentences found in WW-Events:

(5.19) a. During *the three months between the cease-fire and the French referendum on Algeria*, the OAS unleashed a new terrorist campaign.

b. He argued that with *two months of good weather remaining until the onset of the monsoon*, it would be irresponsible to not take advantage of the situation.

c. On *the day after the US's northward crossing of the 38th-parallel border into North Korea*, Mao Zedong ordered the People's Liberation Army's North East Frontier Force to be reorganized into the Chinese People's Volunteer Army, who were to fight the "War to Resist America and Aid Korea".

d. The rate of US combat deaths in Baghdad nearly doubled in *the first seven weeks of the "surge" in security activity.*

e. The Allies withstood *two full days of Persian attacks, including those by the elite Persian Immortals.*

f. *The era of Porfirio Daz's government from 1876-1910* has become known as the Porfiriato.

These examples are amongst the longest and most complex correctly-processed structures in the dataset.

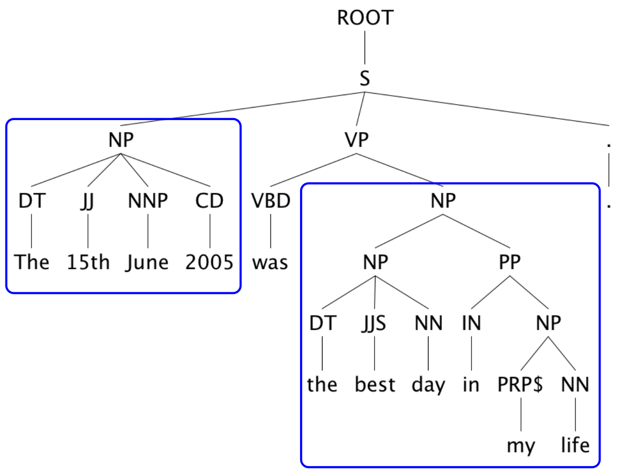


Figure 5.8: A constituency tree for a sentence containing two temporal expressions, one being a subject and one being an object.

The gap between the strict and lenient results is, however, relatively large. An analysis of the errors shows that the majority of these are due to parser errors.This is not surprising, because the datasets used in this experiments contained very complex sentences and even a single incorrectly identified dependency may result in an incorrect extent.

5.5 The Constituency-based Approach

Our second approach to extent recognition using syntactic information is based on phrase structure trees. Here the task is less straightforward than when using dependency trees, because for a given leaf which is a trigger we have a number of phrases that could potentially constitute the corresponding temporal expression, depending on how large a constituent we extract.

As we noted earlier, temporal expressions are arguments to propositions. Therefore our first-cut approach is to choose a phrase which corresponds to a syntactic argument or adjunct to a verb, removing any leading preposition to match the TIMEX2 annotation style. As an example, consider the tree in Figure 5.8; here we have two temporal expressions, *The 15th June 2005* being the subject of the sentence and *the best day in my life* being the object. Note that this approach also correctly handles cases where the temporal expression is part of a verb phrase located deeper in the tree structure, or within a subclause:

(5.20) a. The picture [presents the battle [fought [on [*the 15th December* $_{\mathsf{NP}}$]$_{\mathsf{PP}}$]$_{\mathsf{VP}}$]$_{\mathsf{VP}}$].

b. The general resigned when it became clear that [[*the whole year of efforts* $_{\mathsf{NP}}$] [resulted in no advances $_{\mathsf{VP}}$]$_{\mathsf{S}}$].

5.5.1 The Experimental Set-up

In these experiments we used the following four parsers: the Stanford PCFG Parser[26] (Klein and Manning, 2003), the Bikel Parser[27] (Bikel, 2002) with the Collins emulation settings, the Charniak parser[28] (Charniak, 2000), and the Charniak-Johnson reranking parser[29] (Charniak and Johnson, 2005). All parsers were trained on Sections 02–21 of WSJ Penn Treebank;

In all set-ups, to construct the input data we extract sentences using GATE's Sentence Splitter (based on GATE's Alternate English Tokeniser) and tokenize them with a Penn Treebank tokeniser.[30] We use the same datasets as in the experiments with the dependency-based approach, i.e. the ACE 2005 Training corpus and WikiWars. We also use the same three sets of triggers: word triggers, four digit numbers in the range 1900–2019, and conventionally-formatted dates and times.

5.5.2 The Experiments

5.5.2.1 The First Run

In the first run, we select a candidate temporal expression by starting at a trigger token[31] and working up the syntax tree towards its root until the current node has one of the following nodes as its mother: S, SBAR, SBARQ, SINV, SQ, and FRAG. These are the Penn Treebank II Style (see (Bies et al., 1995)) syntactic tags that can appear at the clause level. When processing a non-sentential fragment such as a section heading, the mother node can also be labelled NP, PP, UCP, X, ADJP, or INTJ.

Results The results of this general approach are shown in Table 5.13 and Table 5.14 for the ACE 2005 Training and WikiWars corpora, respectively.[32] In set-up no. 1, i.e. using only the word triggers, the best result was obtained with the Charniak-Johnson reranking parser for the ACE data (strict F-measure 0.42) and the Bikel/Collins parser for the WikiWars corpus (strict F-measure 0.53). Given the similar lenient F-measure of about 0.80, the method performed much better on the WikiWars data. This can be explained by the higher proportion of well-formed sentence in WikiWars, as discussed earlier. Comparing these results to those obtained with the dependency-based

[26]We used version 1.6.5; see `http://nlp.stanford.edu/software/lex-parser.shtml`.

[27]The parser has been developed at the University of Pennsylvania by Dan Bikel and can be obtained from `http://www.cis.upenn.edu/~dbikel/software.html`. We used version 1.2.

[28]We used the parser version of 05Aug16; see `ftp://ftp.cs.brown.edu/pub/nlparser`.

[29]The parser (we used the version of Aug06) can be obtained from `ftp://ftp.cs.brown.edu/pub/nlparser` and more information is available at `http://bllip.cs.brown.edu/resources.shtml`.

[30]See `http://www.cis.upenn.edu/~treebank/tokenization.html`; we used a slightly modified version (the same as in the experiments with dependency parsers).

[31]In cases where a trigger consists of a few tokens, and therefore also a few syntactic leaf nodes, we start with the last token of the trigger.

[32]We note that the results in consecutive executions with the same settings (i.e. the parser and set-up) differed in the measures of missing, spurious and partially correct annotations but by no more than a few (e.g. three) annotations; the number of correctly recognized expressions remained constant. As this had no impact on the quoted precision, recall and F-measure, the results in the tables simply present one of the executions, and in particular without paying attention to whether this delivered the maximum spurious, missing or partially correct measure. We are not sure where exactly the source of this undeterministic behaviour is, but we suspect it is related to the annotation matching algorithm implemented in the GATE platform which we use for evaluation.

approach, we observe a decrease in performance: the corresponding numbers there were 0.50 and 0.59.

Overall the best results were obtained with set-up #5, which provided strict F-measures of 0.57 and 0.69 for the ACE and WikiWars corpora, respectively. The Charniak-Johnson and Bikel/Collins parsers proved to be the most useful. These results are worse than those obtained with the dependency-based approach, where the corresponding numbers were 0.65 and 0.73; we note that for WikiWars the gap is smaller (0.04) than for the ACE corpus (0.08).

Error Analysis The results with the constituency-based approach are lower than when using dependency trees because in this experimental run we implemented a quite general approach: we select the constituent that is an immediate daughter in the tree of the sentence or verb phrase node. This inevitably generates over-long extents for temporal expressions which are located deeper in the structure of a sentence, as in Example (5.21), or which are modifiers in noun phrases, as in Example (5.22); in both cases this is independent of the location of the noun phrase in the sentence.[33]

(5.21) a. [The initial deployment [of 3,500] [in *March*]] was increased to nearly 200,000...

b. ...was much smaller [than the deployment [in *December*]].

(5.22) a. [Their second *winter* counter-offensive] was a complete failure.

b. They lost [their second *winter* campaign].

In all these sentences the constituents marked with the outermost pairs of square brackets, instead of just the italicised fragments, would be selected as temporal expressions. We will address these issues in the second run of the experiment.

We note, however, that expressions based on the pattern '`the [start|middle|end|...] of <trigger>`', which where problematic for the dependency-based approach, are now handled correctly. The same improvement is observed for date expressions where the syntactic head is the ordinal number, e.g. *the 15th of June 2006.*

Most often the reason for obtaining an incorrect extent is a wrong PP attachment. This concerned all the sentences[34] in Example (5.23) when processed with the Charniak-Johnson parser.

(5.23) a. The Germans launched [an offensive [in *January* $_{\mathsf{PP}}$]$_{\mathsf{NP}}$].

b. The Red Army assisted the partisans in [a joint liberation of the capital city of Belgrade [on *October 20* $_{\mathsf{PP}}$]$_{\mathsf{NP}}$].

c. Lee surrendered [his Army of Northern Virginia [on *April 9, 1865* $_{\mathsf{PP}}$]$_{\mathsf{NP}}$].

d. They surrendered on [[*November 6, 1865*, $_{\mathsf{NP}}$] [in Liverpool, England $_{\mathsf{PP}}$]$_{\mathsf{NP}}$].

In all of the above cases, the prepositional phrases with the temporal expressions were not attached to the verbs but to the noun phrases, as is marked with the bracket notation. In consequence, the whole noun phrases marked with the outermost pairs

[33]The Examples (5.21a) and (5.22a) are sourced from the WikiWars corpus, and Examples (5.21b) and (5.22b) are artificial examples created to demonstrate the issue.

[34]In Example (5.23) we present simplified versions of the sentences; the actual sentences contained more phrases and dependent clauses. The complexity of the original sentences might have influenced the parser; however, this does not change the discussion of the results which we present here.

of square brackets, instead of just the italicised fragments, were annotated as temporal expressions. The Stanford parser, on the other hand, parsed all these sentences correctly and so the extents were also determined correctly.

There are also other errors made by the parsers. For example, the Charniak-Johnson parser incorrectly combined *On 7 August, French and British troops* from the sentence in Example (5.24) into a single prepositional phrase, which resulted in an incorrect extent for the generated annotation.

(5.24) On 7 August, French and British troops invaded the German protectorate of ...

The Bikel/Collins parser, although it made other mistakes in parsing this sentence, provided only *On 7 August* as the PP, which resulted in a correct annotation.

The differences between parsers concern not only the correctness or otherwise of their analyses, but also different treatments in some particular cases. For example, for the phrase *defeated from October to March* the Stanford parser produces the structure as in Example (5.25a), but the Charniak-Johnson parser provides the more complex structure of hierarchical prepositional phrases shown in Example (5.25b).

(5.25) a. [defeated [from [October $_{\text{NP}}$] $_{\text{PP}}$] [to [March $_{\text{NP}}$] $_{\text{PP}}$] $_{\text{VP}}$]

b. [defeated [[from [October $_{\text{NP}}$] $_{\text{PP}}$] [to [March $_{\text{NP}}$] $_{\text{PP}}$] $_{\text{PP}}$] $_{\text{VP}}$]

With the current method of processing the syntax trees, this results in correct annotations with the Stanford parser, but in an over-long extent with the Charniak-Johnson parser. To handle phenomena like this one could introduce a more parser-specific approach.

5.5.2.2 The Second Run

In the second run of the experiment we added three additional stopping conditions for the process of working up the syntax tree:

- To handle temporal expressions that appear as modifiers within noun phrases (as, for example, in *the winter offensive*), we detect whether the noun phrase contains any other nouns following the trigger, and in such a case we annotate only the trigger. This does not guarantee the correct extent in all cases, because the temporal expression may contain some additional tokens (i.e. modifiers of the trigger), but it is still a good heuristic.

- To detect whether an expression is a prepositional phrase within a noun phrase (see Example (5.21)), we annotate only the PP, not the whole NP as was done in the first run.

- To annotate temporal expressions provided in parentheses, as in the sentence below, we stop at any node which is the daughter of a node labelled ‘PRN’:

  (5.26) They proceeded to swear the Tennis Court Oath (*20 June 1789*).

We also removed the SBAR node as a stopping condition because we observed that these constituents are sometimes parts of temporal expressions with complex syntactic structure.[35]

[35]In the evaluations it turned out that this change improves the recognition of only four expressions in WikiWars, and spoiled recognition of three expressions in the ACE corpus.

Set-up		Measure	Stanford	Bikel/Collins	Charniak	Char.-Johnson
1	(Words)	Correct	1754	1928	1874	1991
		Missing	1635	1621	1624	1613
		Spurious	300	400	336	335
		Part. Correct	2039	1879	1930	1824
		Strict P/R/F	0.43/0.32/0.37	0.46/0.36/0.40	0.45/0.35/0.39	0.48/0.37/0.42
		Lenient P/R/F	0.93/0.70/0.80	0.90/0.70/0.79	0.92/0.70/0.80	0.92/0.70/0.80
2	(Dates)	Correct	63	79	59	167
		Missing	4351	4353	4352	4353
		Spurious	10	69	69	68
		Part. Correct	1014	996	1017	908
		Strict P/R/F	0.06/0.01/0.02	0.07/0.01/0.02	0.05/0.01/0.02	0.15/0.03/0.05
		Lenient P/R/F	0.99/0.20/0.33	0.94/0.20/0.33	0.94/0.20/0.33	0.94/0.20/0.33
3	(Years)	Correct	132	165	153	275
		Missing	4325	4328	4323	4323
		Spurious	8	14	9	9
		Part. Correct	971	935	952	830
		Strict P/R/F	0.12/0.02/0.04	0.15/0.03/0.05	0.14/0.03/0.05	0.25/0.05/0.08
		Lenient P/R/F	0.99/0.20/0.34	0.99/0.20/0.34	0.99/0.20/0.34	0.99/0.20/0.34
5	(Words) +Year+Dates	Correct	2775	2945	2897	3012
		Missing	676	662	669	655
		Spurious	382	376	373	373
		Part. Correct	1977	1821	1862	1761
		Strict P/R/F	0.54/0.51/0.53	0.57/0.54/0.56	0.56/0.53/0.55	0.59/0.55/0.57
		Lenient P/R/F	0.93/0.88/0.90	0.93/0.88/0.90	0.93/0.88/0.90	0.93/0.88/0.90

Table 5.13: The results for extent recognition obtained with a constituency-based tagger on the ACE 2005 Training corpus: the first run.

Set-up		Measure	Stanford	Bikel/Collins	Charniak	Char.-Johnson
1	(Words)	Correct	958	1216	1099	1131
		Missing	840	841	851	855
		Spurious	27	26	26	27
		Part. Correct	883	624	731	695
		Strict P/R/F	0.51/0.36/0.42	0.65/0.45/0.53	0.59/0.41/0.48	0.61/0.42/0.50
		Lenient P/R/F	0.99/0.69/0.81	0.99/0.69/0.81	0.99/0.68/0.81	0.99/0.68/0.81
2	(Dates)	Correct	356	531	434	454
		Missing	1888	1897	1890	1895
		Spurious	0	0	0	0
		Part. Correct	437	253	357	332
		Strict P/R/F	0.45/0.13/0.20	0.68/0.20/0.31	0.55/0.16/0.25	0.58/0.17/0.26
		Lenient P/R/F	1.00/0.30/0.46	1.00/0.29/0.45	1.00/0.30/0.46	1.00/0.29/0.45
3	(Years)	Correct	427	523	480	492
		Missing	1827	1833	1827	1829
		Spurious	0	0	0	0
		Part. Correct	427	325	374	360
		Strict P/R/F	0.50/0.16/0.24	0.62/0.20/0.30	0.56/0.18/0.27	0.58/0.18/0.28
		Lenient P/R/F	1.00/0.32/0.48	1.00/0.32/0.48	1.00/0.32/0.48	1.00/0.32/0.48
5	(Words) +Year+Dates	Correct	1703	1790	1768	1779
		Missing	345	345	349	345
		Spurious	164	167	163	166
		Part. Correct	633	546	564	557
		Strict P/R/F	0.68/0.64/0.66	0.72/0.67/0.69	0.71/0.66/0.68	0.71/0.66/0.69
		Lenient P/R/F	0.93/0.87/0.90	0.93/0.87/0.90	0.93/0.87/0.90	0.93/0.87/0.90

Table 5.14: The results for extent recognition obtained with a constituency-based tagger on WikiWars: the first run.

Set-up		Measure	Stanford	Bikel	Charniak	Char.-Johnson
1	(Words)	Correct	2344	2468	2385	2468
		Missing	1574	1568	1569	1568
		Spurious	447	448	443	446
		Part. Correct	1510	1392	1474	1392
		Strict P/R/F	0.54/0.43/0.48	0.57/0.45/0.51	0.55/0.44/0.49	0.57/0.45/0.51
		Lenient P/R/F	0.90/0.71/0.79	0.90/0.71/0.79	0.90/0.71/0.79	0.90/0.71/0.79
2	(Dates)	Correct	362	409	333	451
		Missing	4346	4347	4348	4348
		Spurious	69	69	69	69
		Part. Correct	720	672	747	629
		Strict P/R/F	0.31/0.07/0.11	0.36/0.08/0.12	0.29/0.06/0.10	0.39/0.08/0.14
		Lenient P/R/F	0.94/0.20/0.33	0.94/0.20/0.33	0.94/0.20/0.33	0.94/0.20/0.33
3	(Years)	Correct	245	250	245	241
		Missing	4316	4312	4314	4314
		Spurious	41	40	41	41
		Part. Correct	867	866	869	783
		Strict P/R/F	0.21/0.05/0.07	0.22/0.05/0.08	0.21/0.05/0.07	0.21/0.04/0.07
		Lenient P/R/F	0.96/0.20/0.34	0.97/0.21/0.34	0.96/0.21/0.34	0.96/0.21/0.34
5	(Words) +Year+Dates	Correct	3353	3484	3398	3484
		Missing	621	613	614	614
		Spurious	413	409	405	411
		Part. Correct	1454	1331	1416	1330
		Strict P/R/F	0.64/0.62/0.63	0.67/0.64/0.65	0.65/0.63/0.64	0.67/0.64/0.65
		Lenient P/R/F	0.92/0.89/0.90	0.92/0.89/0.90	0.92/0.89/0.90	0.92/0.89/0.90

Table 5.15: The results for extent recognition obtained with a constituency-based tagger on the ACE 2005 Training corpus: the second run.

Set-up		Measure	Stanford	Bikel	Charniak	Char.-Johnson
1	(Words)	Correct	1243	1513	1365	1379
		Missing	763	768	770	766
		Spurious	32	26	31	29
		Part. Correct	675	400	546	536
		Strict P/R/F	0.64/0.46/0.54	0.78/0.56/0.65	0.70/0.51/0.59	0.71/0.51/0.60
		Lenient P/R/F	0.98/0.72/0.83	0.99/0.71/0.83	0.98/0.71/0.83	0.99/0.71/0.83
2	(Dates)	Correct	475	684	684	604
		Missing	1858	1858	1858	1858
		Spurious	0	0	0	0
		Part. Correct	348	139	139	219
		Strict P/R/F	0.58/0.18/0.27	0.83/0.26/0.39	0.83/0.26/0.39	0.73/0.23/0.34
		Lenient P/R/F	1.00/0.31/0.47	1.00/0.31/0.47	1.00/0.31/0.47	1.00/0.31/0.47
3	(Years)	Correct	574	659	629	631
		Missing	1804	1802	1808	1803
		Spurious	0	0	0	0
		Part. Correct	303	220	244	247
		Strict P/R/F	0.65/0.21/0.32	0.75/0.25/0.37	0.72/0.23/0.35	0.72/0.24/0.35
		Lenient P/R/F	1.00/0.33/0.49	1.00/0.33/0.49	1.00/0.33/0.49	1.00/0.33/0.49
5	(Words) +Year+Dates	Correct	1912	1992	1970	1968
		Missing	326	328	328	328
		Spurious	170	172	171	171
		Part. Correct	443	361	383	385
		Strict P/R/F	0.76/0.71/0.73	0.79/0.74/0.77	0.78/0.73/0.76	0.78/0.73/0.76
		Lenient P/R/F	0.93/0.88/0.90	0.93/0.88/0.90	0.93/0.88/0.90	0.93/0.88/0.90

Table 5.16: The results for extent recognition obtained with a constituency-based tagger on WikiWars: the second run.

Finally, just as in the experiments with dependency parsers, we now do postprocessing to remove the trailing commas from the extent and additionally we also remove quote characters if these occur as the first or last character of the extent (this problem did not occur with the dependency-based approach); this improved the recognition of ten expressions in WikiWars, and two expressions in the ACE corpus.

The Results The results of the second run are shown in Table 5.15 and Table 5.16 for the ACE 2005 Training and WikiWars corpora, respectively. The best results on both corpora were obtained using the Bikel/Collins parser; however, for the ACE corpus the Charniak-Johnson parser was just as useful (for both set-ups no. 1 and no. 5 these two parsers provided exactly the same number of correctly recognized expressions).

We observe a significant improvement over the results from the first run (see Tables 5.13 and 5.14): about 0.07–0.12 in strict F-measure in most set-ups. Compared to the dependency-based approach, the strict results are comparable for the ACE corpus, and better for WikiWars with 0.06 improvement in strict F-measure for set-up no. 1 and 0.04 improvement for set-up no. 5. As expected the lenient results remained the same (with the exception only of a small drop for year triggers only, i.e. set-up no. 3 for WikiWars).

Given that we use the same sets of triggers as in the experiments with using the functional dependencies, the gain from applying the syntactic method compared to the performance of recognizing triggers only has also improved for WikiWars: for set-up no. 1 the gain is now 0.50 (was 0.44), and for set-up no. 5 the gain is 0.22 (was 0.18).

Error Analysis When analysing the expressions that were not recognized correctly, we discovered a number of different underlying reasons for these errors.

First, as noticed earlier, the parsers make mistakes. For example, the Bikel/Collins parser included *six days later* in the subject noun phrase in the following sentence, rather than analysing it as an adjunct:

(5.27) *[*Six days later* this regime $_{\mathsf{NP}}$] was replaced ...

Additionally, in the ACE corpus parsing was more challenging because of the sometimes problematic formatting of the text, with no clear sentence boundaries in many cases.

While the method successfully recognizes expressions for which the dependency-based approach failed because a trigger was not the head of the expression (for example, *the beginning of next week*), some additional semantic distinctions are necessary to distinguish between these expressions and expressions like those in Example (5.28):

(5.28) a. several international agreements of *the past two centuries*

b. their response of *14 January*

c. the course of *a few hours*

Currently, in all these cases the method generates over-long extents, encompassing the entire phrases shown rather than just the italicised elements.

In TIMEX2, phrases containing conjunctions are annotated with multiple temporal expressions. However, as we noted in Chapter 4, this may be not the best solution; there, we suggested that each such phrase might be considered as a single set expression. In Figure 5.9 we present a syntactic tree with the expression found by the constituency-based approach marked with a blue box. In TIMEX2 the expected expressions are *the spring* and *early summer of 1943*. Not only is one of the noun phrases split between

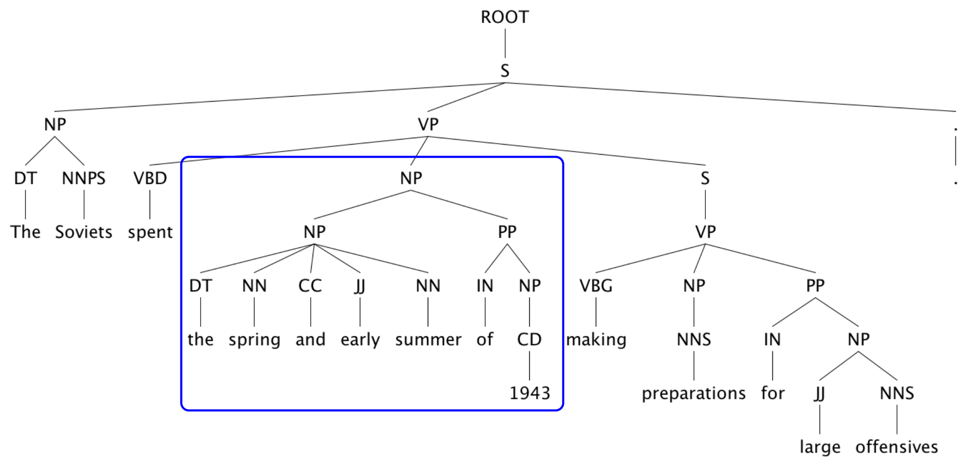


Figure 5.9: A constituency tree for a sentence containing two temporal expressions in a conjunction structure with an ellipsis.

two expressions, the second expression spans over a part of one noun phrase and a separate prepositional phrase. Addressing this issue requires a special rule that would detect the use of conjunction between two triggers in a single syntactic node and split the annotation accordingly.

While some expressions undeniably require the use of patterns in order to be correctly recognized (e.g. formatted dates such as *16-08-2007*), the patterns we used in the experiment were not ideal, and therefore some expressions (e.g. *early 1936*; *March 31, 1936*) were recognized only partially. Also, the application of the rules that produced final annotations blocked correct recognition of full extents via the syntactic method, e.g. in the case of *the end of January 1936* where *January 1936* was recognized by a rule as an expression. We also noticed that, when processing with the Bikel/Collins parser in set-up no. 1, where no such patterns were used, date expressions like *9 November* or *November 20, 1953* or dates with time information *May 8 1:00AM* triggered by a month name were not recognized correctly, resulting in extents that were too short.

5.5.3 Extent Recognition of Event-based Expressions

Just as with the experiments using the dependency-based approach, we now evaluate the constituency-based approach on data containing event-based expressions only. We use the same two datasets as previously: TG-Events containing 25 expressions found in the TIMEX2 guidelines, and WW-Events containing 122 event-based expressions sourced from the WikiWars corpus. We run the experiments using the extended set of word triggers and the configurations used in the second run of the method applied to the larger corpora.

The results are presented in Tables 5.17 and 5.18. For both datasets the best results are obtained with the Charniak parser and are practically the same: 0.84 and 0.83 (precision, recall and F-measure are again the same). This is not only a much better result than is achieved by using pattern-based taggers that do not make use of syntax, but also a significant improvement over the dependency-based approach for the WW-Events dataset, where the corresponding results were 0.84 and 0.69.

The sources of errors are much the same as those identified earlier: the PP attach-

					Strict			Lenient		
Method	Corr.	Miss.	Spur.	Part.	Prec.	Recall	F	Prec.	Recall	F
Word-triggers	0	0	0	25	0.00	0.00	0.00	1.00	1.00	1.00
Word-trig./Stanford	13	0	0	12	0.52	0.52	0.52	1.00	1.00	1.00
Word-trig./BikelCollins	18	0	0	7	0.72	0.72	0.72	1.00	1.00	1.00
Word-trig./Charniak	21	0	0	4	0.84	0.84	0.84	1.00	1.00	1.00
Word-trig./Ch.-John.	20	0	0	5	0.80	0.80	0.80	1.00	1.00	1.00
DANTE	2	0	1	23	0.08	0.08	0.08	0.96	1.00	0.98
TERSEO	1	3	0	21	0.05	0.04	0.04	1.00	0.88	0.94

Table 5.17: The results for extent recognition for event-based expressions in TG-Events using the constituency-based approach (the second run) and pattern-based systems.

					Strict			Lenient		
Method	Corr.	Miss.	Spur.	Part.	Prec.	Recall	F	Prec.	Recall	F
Word-triggers	0	0	1	122	0.00	0.00	0.00	0.99	1.00	1.00
Word-trig./Stanford	84	1	0	37	0.69	0.69	0.69	1.00	0.99	1.00
Word-trig./BikelCollins	97	0	0	25	0.80	0.80	0.80	1.00	1.00	1.00
Word-trig./Charniak	101	0	0	21	0.83	0.83	0.83	1.00	1.00	1.00
Word-trig./Ch.-John.	98	0	0	24	0.80	0.80	0.80	1.00	1.00	1.00
DANTE	7	11	2	104	0.06	0.06	0.06	0.98	0.91	0.94
TERSEO	3	17	2	102	0.03	0.02	0.03	0.98	0.86	0.92

Table 5.18: The results for extent recognition for event-based expressions in WW-Events using the constituency-based approach (the second run) and pattern-based systems.

ment problem and other parsing errors; the presence of conjunctions; and the need for semantic distinctions.

5.6 Conclusions

In this chapter we have presented a new approach to the recognition of temporal expressions, using syntactic information to grow extents around already-identified trigger terms.

We experimented with two types of syntactic analysis: functional dependencies between tokens and hierarchical syntactic constituents. In each case we used four different parsers: Minipar, Connexor and C&C provided us with dependency trees, and the Bikel/ Collins, Charniak and Charniak-Johnson parsers produced phrase structure trees, and the Stanford parser was used for both types of syntactic information.

The results proved that both dependencies and syntactic constituents can be successfully used to recognize the extents of temporal expressions in texts. The approach with dependencies is more straightforward to implement, because if we assume that the trigger is the syntactic head of the expression, then the extent is determined by the tokens found in the dependency tree rooted by the trigger. When using phrase structures, we have to select from amongst the multiple constituents that contain the trigger. However, it turned out that by implementing just a few parser-independent selection rules we managed to get in our experiments the same or slightly better recognition results with this approach than by using dependencies. This, of course, does not necessarily prove the advantage of one approach over the other in the general case, since different parsers were used, and it is not out of the question that some of the problematic cases could be addressed in the dependency-based approach if we did not always assume that the trigger is the root of the selected subtree.

Overall, the best results obtained for the recognition of temporal expressions with the constituency-based approach were 0.51 F-measure for the ACE 2005 Training corpus, and 0.65 for the WikiWars corpus, where the upper bounds determined by the detection results were 0.79 and 0.83, respectively. With the additional application of date and year recognition rules that produced final annotations, the results grew to 0.65 for the ACE corpus and 0.77 for WikiWars, with the upper bounds for both corpora being 0.90.

The results are especially good for the WikiWars corpus. The difference in the performance on the two corpora is related to problems with correct sentence segmentation and parsing in the ACE documents. This suggests that for domains or text types where sentences may be 'broken' (e.g. speech transcripts, blogs, chat logs, notes) approaches based on syntactic information will not perform well. In such cases it may be more appropriate to use methods where we just analyse the lexical items occurring in a limited textual context around the trigger.

On the other hand, the two approaches that we have tested seem to be the best choice for recognition of event-based expressions. Here the best strict F-measure achieved is 0.84 for the event-based expressions found in the TIMEX2 guidelines, and 0.83 for those found in the WikiWars corpus. For comparison, the two pattern-based systems we tried achieved results in the range 0.03–0.06. An open question remains about how to identify the occurrence of an event-based expression so that we know when to apply either of the two proposed syntax-based approaches. One possibility is to perform independent event-recognition processing, and then to test for the presence of the trigger and an event in the same syntactic subtree. This of course requires new experiments which we leave as future work.

We identified a number of sources of error; these include incorrect sentence splitting, and errors in parsing, with wrong PP attachment being a common problem. In the case of dependencies, we observed differences between the parsers with respect to which token is considered the head in a given relation; but we also observed that not all temporal expressions in TIMEX2 are headed by triggers. We note that some types of expressions (e.g. ranges and conjoined expressions) could be correctly recognized, and the results correspondingly improved, by making use of more detailed rules adjusted to the specific outputs of particular syntactic parsers.

Chapter 6

The Interpretation of Temporal Expressions

In the previous chapters, we discussed the different kinds of temporal expressions and their recognition in text; in this chapter we focus on their semantics.

In Section 6.1 we introduce the interpretation task, discuss what we call the local and global semantics of temporal expressions and, taking TIMEX2 as a starting point, we introduce the ideas behind our string-based representation of semantics. Then, in Section 6.2 we present our representation of local semantics, called LTIMEX, which is designed for easy integration with the existing annotation schemes.

Section 6.3 deals with the problem of finding the reference time for the interpretation of context-dependent temporal expressions. For some documents, e.g. news stories, using the document time-stamp as the reference time is a good heuristic; but for other text types this is an oversimplification that leads to the incorrect interpretation of expressions. Since this is a very important issue for broad-coverage processing of temporal expressions, we review the work done in this area and experiment with a number of approaches.

In Section 6.4 we study the interpretation of bare weekday names such as *Tuesday.* These expressions require special treatment because their underspecification is resolved in a different way from other expressions. We provide a detailed comparison of the various approaches to the problem, employing re-implementations of key techniques from the literature and a range of additional heuristic-based approaches.

Although at first glance it may seem that the interpretation step can be carried out simply either by calculating the offset from some reference time, or by filling in the missing elements in the representation of underspecified expressions, it turns out that there are various problematic cases that require more complex solutions. We discuss these cases in Section 6.5.

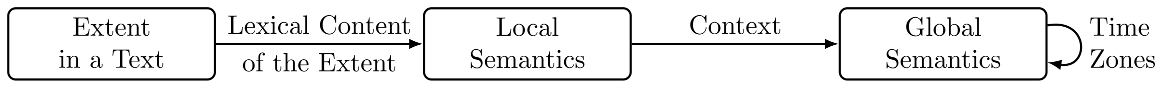


Figure 6.1: The interpretation task.

6.1 The Interpretation Task

The interpretation task is the stage of processing a temporal expression in which we generate a meaning representation for the expression. In our work we explicitly distinguish between what we call the **local** and the **global** semantics of an expression.

The local semantics is the result of an analysis of the meaning of a temporal expression without any context being taken into account.[1] At this level only the tokens within the extent of the expression are considered when building the semantic representation.

The global semantics represents the meaning of an expression taking into account the document context and, in some cases, also world knowledge (e.g. the dates of well-known events). This is the level at which we create interpretations corresponding to those found in TIMEX2 and TimeML. One can also go one step further, and normalize all the temporal values to a single chosen time zone, e.g. GMT. This has significance for processing documents published by different sources in different parts of the world.[2]

The full interpretation process involves generation of both levels of semantic representation. First the local semantics is obtained by parsing and analysing the structure of the expression's extent; then the global semantics is derived from the local semantics by adding information derived from the context, and, additionally, geo-temporal information about time zones may be applied to generate the normalised semantic values; this is illustrated in Figure 6.1. In the work described here, we are generally concerned with the process of deriving the global semantics without the final time zone normalisation step.

6.1.1 Local and Global Semantics

Local semantics is the term that we use to refer to the meaning of temporal expressions independent of any contextual information: the representation of local semantics reveals only the meaning of the tokens (mainly words, but also numbers and punctuation marks) of which the expression is built. Of course, for context-independent expressions, such as those in Example (6.1), the local semantics is no different from the **global semantics**, i.e. the meaning of the temporal expression in the context of the document in which it appears.

(6.1) a. I was born on *14th November 1978.*

[1]This is not entirely true, since at least a particular calendar and a particular time zone are assumed, so values will always be relative to these. For most practical purposes, however, this makes the expressions context-free.

[2]As an example of a scenario where such processing would be useful, consider the need to carry out an analysis of events happening in a chosen geographic region based on news media articles published in countries around the world. A system like JRC's NewsExplorer (see `http://emm.newsexplorer.eu`) is an example of an application which benefits from automatic processing of temporal information that involves this kind of normalisation.

b. I was born in *the 1980s.*

For context-dependent expressions, such as those in Example (6.2), local semantics does not reveal which specific temporal entity is referred to by the expression, but, instead, either represents underspecified temporal information (as in Example (6.2a)), or provides a recipe to calculate the temporal location of the referred-to entity (as in Example (6.2b)).

(6.2) a. I was born on *the 14th November.*

b. I will call you *tomorrow.*

In either case the unambiguous reference to the actual temporal entity can be obtained only when the context is known.

The key principle here is that the local semantics of the expression is always the same, no matter what the context is. For example, *November* always refers to 'the 11th month of a calendar year'; *tomorrow* always means 'the day following the day of the utterance date'. The global semantics, on the other hand, may be different depending on the context in which the expression is used: in one document *tomorrow* may be used to mean a different date than in another document.

The global semantics is obtained from the local semantics either by providing the missing information in the case of an underspecified expression, or by calculating the offset with respect to some reference time found in the context. For example, deriving the global semantics for the temporal expression in Example (6.2a) requires providing the value for the year, and for the expression in Example (6.2) we need to add one day to the date of the speech time.

For underspecified expressions, the interpretation process obeys what we call **the principle of full specification**: information about all temporal units coarser than the most fine-grained one provided in the local semantics must be found. For example, at the level of local semantics the expression *5pm on a January day* specifies a month and an hour, so when generating the global semantics all missing slots coarser than the hour—i.e., day and year—must be resolved based on the reference time; but there is no requirement to determine values finer-grained than the hour.[3] Note, however, that the values that get filled in are not necessarily the values carried by the reference time; consider the following example:

(6.3) We are flying to Fiji in *February.*

If the above sentence is uttered in October of year 2004, the value that must be provided for the global semantics of *February* is 2005.

Finding the correct reference time plays a significant role in the interpretation step. Depending on the expression type and the context of use, the expression may be relative to (a) the time at which the utterance was made, (b) another expression occurring in the text, or (c) the time of some event mentioned in the document. In the first case, the reference time is 'the time at which the assertion in the text is made'; depending on the source of the document, this may be the publication or modification date of a document (for example, in the case of news stories), the time of creation of a file in a filesystem, or the time of sending an email or making a phone-call. To generalize across these different possibilities, in our work we will refer to this point as

[3]Of course, for non-specific expressions (e.g. those used generically) some slots may remain unspecified.

the **document creation date** (DCD), or **document time-stamp**.[4] The expressions that are interpreted using the document time-stamp are, for example, deictic offsets such as *tomorrow* or *two months ago*. The second case requires tracking the temporal focus throughout the discourse, which may not be straightforward to do. Expressions that require temporal focus tracking are, for example, anaphoric offsets, such as *the following day* or *two months earlier*. Underspecified expressions are in a way special since, depending on the context of use, they may need to be interpreted either with respect to the document time-stamp or another expression appearing in the content of the document; unlike in the case of an offset expression, the tokens found in the extent of an underspecified expression do not provide information about which reference time to use. Finally, the third case requires us to carry out event extraction and time-stamping.[5] Expressions that rely on event time-stamping are not only those that are event-based (e.g. *the third day of the battle*), but also offsets from mentioned events such as *two days later* in the following example:

(6.4) With the arrival of more British troops the city surrendered. *Two days later* the peace treaty was signed.

However, the processing of event references is beyond the scope of the work presented here.

6.1.2 The Representation of Global Semantics in TIMEX2

One approach to representing the meaning of temporal expressions is to use a string-based encoding of the global semantics; this is how temporal values are expressed in both TIMEX2 and TimeML. Both these annotation schemes were developed by the information extraction community and are widely-used standards for representing the values of temporal expressions in text; they serve as target representations for temporal expression taggers.

TIMEX2 defines five attributes to represent the meaning of a temporal expression: `VAL`, `ANCHOR_VAL`, `ANCHOR_DIR`, `MOD`, and `SET`. These attributes are used, respectively, to encode the temporal location of a point on a timeline or a duration of a period; to encode the temporal location of one of the period's end-points; to capture the direction of temporal reference from the anchoring point; to express modifications to more basic temporal values; and to flag whether the temporal expression refers to a set of temporal entities.

Values of the `VAL` and `ANCHOR_VAL` attributes use a string representation based on formats defined in the ISO-8601 standard: calendar date (`YYYY-MM-DD`), week date (`YYYY-Www-D`), time of day (`hh:mm:ss`), date and time (`YYYY-MM-DDThh:mm:ss`), and duration (`PnYnMnDTnHnMnS` or `PnW`). The individual character positions in the date and time strings correspond to particular granularities of temporal information, as demonstrated by the examples in Table 6.1. TIMEX2 extends the encodings provided in the ISO standard by introducing tokens representing additional temporal granularities: for example, in place of a month number, TIMEX2 also permits codes for year seasons (e.g. `SU` for summer), half-years (e.g. `H1`), quarter-years (e.g. `Q3`); in place of the hour, a code of a part of a day can be provided (e.g. `MO` for morning). It also added support for BCE years, references to the distant past (i.e. billion, million, thousand years ago) and

[4]Since we deal primarily with textual examples, this is a reasonable name to use; it is less intuitive for spoken language, of course, for which 'speech time' would be more appropriate.

[5]Note that event time-stamping relies on the processing of temporal expressions.

Attribute Value	Meaning	Attribute Value	Meaning
1	The second millennium AD	P2Y	2 years
19	The 20th century AD	P2Y3M	2 years and 3 months
199	The 1990s	P3W	3 weeks
1992	Year 1992	P9D	9 days
1992-06-27	27 June 1992	P2DT6H	2 days and 6 hours
1992-06-27T18:04	27 June 1992 18:04	PT8H	8 hours
1992-06-27T18:04:56	1992-06-27 18:04:56	PT12M	12 minutes
1992-06-27TMO	morning of 27 June 1992	PT8H12M	8 hours and 12 minutes
1992-W04	The fourth week of 1992	PT3.5H	3.5 hours
1992-SU	Summer of 1992	P2DE	2 decades
1992-H1	1st half of 1992	P3CE	3 centuries
1992-Q3	3rd quarter of 1992		
BC0346	Year 346BC		
MA6	6 million years ago		
PAST_REF	vague reference to past		

Table 6.1: Examples of value encoding in TIMEX2 for points and periods.

general references to past, present and future. For non-specific uses of expressions (as in, for example, *a sunny day in June*) TIMEX2 uses an uppercase `X` to fill the slots at the unspecified granularities (here, `XXXX-06-XX`). In regard to the encoding of duration, TIMEX2 adds new temporal units for decades (`PnDE`), centuries (`PnCE`) and millennia (`PnML`).

In documents, a temporal expression tagger encodes these values as attributes of inline XML annotations. In the following examples, we assume the sentences were uttered at 4:17 pm on Friday 10th July 2009:

(6.5) a. I left town on `<TIMEX2 VAL="2008-07-15">`*15th July 2008*`</TIMEX2>`.

b. Let's meet again on `<TIMEX2 VAL="2009-07-20">`*the twentieth*`</TIMEX2>`.

c. I will call you on `<TIMEX2 VAL="2009-07-13">`*Monday*`</TIMEX2>`.

d. I met him `<TIMEX2 VAL="2006">`*three years ago*`</TIMEX2>`.

e. Our company doubled its profit in `<TIMEX2 VAL="P4M" ANCHOR_DIR="ENDING" ANCHOR_VAL="2008-12">`*the last four months of 2008*`</TIMEX2>`.

Such a flat representation of meaning is particularly useful for inline annotation schemes because a single string can be used as a value of one attribute. What is notable about the temporal expression in Example (6.5a) is that it is explicit (i.e. fully-specified): the temporal value can be computed using the lexical content of the string alone, without any reference to context. Not all temporal expressions are of this kind; rather, many are context-dependent (as in Examples (6.5b)–(6.5e)), in that they only partially specify a temporal value, and require incorporation of information available in the context in order to derive their full interpretation. Unfortunately, the nature of the TIMEX2 and TimeML representation means that there is no way of annotating the value of temporal expressions until this full interpretation has been carried out.

Attribute	Comments
`L-VAL`	Encodes the local semantics of expressions concerning the temporal location of points: for underspecified values, the missing slots are filled with `x`; underspecified time is separated from the date components with `t`; for offsets, the encodings start with `+`, `-`, `>` or `<`; ordinally-specified elements are encoded with the pattern `$nu`. Encoding of durations is the same as in TIMEX2.
`L-ANCHOR_VAL`	Local semantics of temporal location of points; see the description of `L-VAL`.
`L-ANCHOR_DIR`	Same values as for the `ANCHOR_DIR` attribute in TIMEX2
`L-MOD`	Same values as for the `MOD` attribute in TIMEX2
`L-SET`	Same values as for the `SET` attribute in TIMEX2
`L-TYPE`	Encodes the taxonomical type of an expression. Possible values: `EXPLICIT`, `UNDERSPECIFIED`, `OFFSET`, `OFFSET-DEICTIC`, `OFFSET-ANAPHORIC`, `EVENT-BASED`, `EVENT-BASED-POINT`, `EVENT-BASED-PERIOD`, `PERIOD`, `SET`, `SET-POINTS`, `SET-PERIODS`
`L-EVENT_ID`	Stores an identifier of an event for event-based expressions.
`L-ANCHOR_TYPE`	For anchored period expressions where the anchor is an offset, indicates the type of the offset; the possible values of the attribute are `DEICTIC` and `ANAPHORIC`.

Table 6.2: A summary of the attributes of LTIMEX.

6.2 LTIMEX: A String-based Representation of Local Semantics

TIMEX2 was designed to allow the annotation of temporal expressions with what we are calling their global semantics, i.e. the temporal value obtained by interpreting the expression in the context of the document in which it is used. Our experience with the development of a temporal expression tagger (first presented in (Mazur and Dale, 2007)) revealed that it is beneficial for both design and evaluation to explicitly recognize the semantics of the expression with no context involved; as discussed above, we refer to this as the local semantics, representing the partial and underspecified context-free meaning of expressions.

In this section we provide a detailed description of our string-based representation of the local semantics of temporal expressions, which we call **LTIMEX**. It is designed to be compatible with existing annotation schemes, especially TIMEX2. This compatibility has two purposes: (i) it is human-readable and requires minimum effort to use, especially for an annotator already familiar with TIMEX2; (ii) it provides a relatively easy means of conversion from local semantics to global semantics, which in our work is represented by the standard TIMEX2 notation; but our encoding can also be combined with TimeML.

LTIMEX extends the set of attributes from TIMEX2 to provide a vocabulary for capturing partially specified meaning. Some of these local attributes mirror the existing attributes completely; others, however, add new types of information that are intended to be of use to a subsequent processing stage that determines the global semantics of the temporal expression. The attributes provided by LTIMEX are shown in Table 6.2. The

Expression Type	Example Expression
Explicit Point	*Friday, 3 April 1998*
Underspecified Point	*23rd June*
Deictic Offset	*tomorrow*
Anaphoric Offset	*the next month*
Event-based Point	*the day when the last fortress fell*
Duration	*six months and two days*
Event-based Duration	*the first two minutes of the meeting*
Ordinally-specified	*the last Tuesday in 1997*
Modified Points	*the middle of August*
Modified Durations	*nearly two decades*
Non-specific Point	*a sunny day in July*
Set	*every Tuesday*

Table 6.3: The types of temporal expressions.

final annotation of an expression has then both the attributes provided by LTIMEX and those provided by TIMEX2, as shown in the following example:[6]

(6.6) `<TIMEX2 L-TYPE="OFFSET-DEICTIC" L-VAL="+0000-00-00" VAL="2010-07-15">` *Today*`</TIMEX2>`, I woke up at `<TIMEX2 L-TYPE="UNDERSPECIFIED" L-VAL="xxxx-xx-xxt10:00" VAL="2010-07-15T10:00">`*ten o'clock*`</TIMEX2>`.

A key feature here is the `L-TYPE` attribute, which stores the type of an expression; this captures the essential distinctions between different kinds of expressions, so that this information can be used to guide subsequent processing. In Table 6.3 we present the types of temporal expressions that we distinguish in this work, along with examples of each. These do not represent a flat taxonomy: the major types are expressions referring to single point and duration temporal entities, each of which may have subtypes; but there are also expressions referring to sets of such temporal entities, as well as ordinally-specified, modified and non-specific expressions; we discussed these distinctions when presenting our taxonomy in Chapter 4. The LTIMEX representations for all of these types are discussed below.

6.2.1 Explicit Expressions

These expressions are the only context-independent point expressions. For these, the local semantics is always the same as the global semantics, so our `L-VAL` simply mirrors the `VAL` in TIMEX2. We present some examples in Table 6.4.

If an expression refers to a date (of day granularity) then any accompanying information about the weekday is ignored, just as is the case in TIMEX2; this is shown in Row 2 of the table. Dates in the US format (Row 4) must be properly distinguished from the European format (Row 3), but the semantic value is of course independent of the format used. If there is any time information expressed with the date (Row 7), then in the semantic representation it is separated using the `T` character.

[6]Assume the utterance shown here was made on 15th July 2010.

No	Expression	Representation (`L-VAL`)
1	3rd January 1987	1987-01-03
2	Friday, 3 April 1998	1998-04-03
3	24/03/1980	1980-03-24
4	03/24/1980	1980-03-24
5	November 1996	1996-11
6	1960s	196
7	12th January 2001 11:59 pm	2001-01-12T23:59

Table 6.4: Examples of explicit dates and times expressed in LTIMEX.

No	Expression	Representation (`L-VAL`)
1	January 3	xxxx-01-03
2	the nineteenth	xxxx-xx-19
3	November	xxxx-11
4	summer	xxxx-SU
5	'63	xx63
6	the '60s	xx6
7	9 pm	xxxx-xx-xxT21
8	11:59 pm	xxxx-xx-xxT23:59
9	eleven in the morning	xxxx-xx-xxT11:00
10	ten minutes to 3	xxxx-xx-xxt02:50
11	15 minutes after the hour	xxxx-xx-xxtxx:15
12	Friday	xxxx-Wxx-5
13	8:00 p.m. Friday	xxxx-Wxx-5T20:00
14	eight o'clock Friday	xxxx-Wxx-5t08:00

Table 6.5: Examples of underspecified expressions in LTIMEX.

6.2.2 Underspecified Expressions

Underspecified expressions differ from explicit expressions in that they omit elements of information, which then have to be recovered from the context by some process of interpretation. LTIMEX provides for the representation of underspecified expressions by marking those elements of the temporal value that are missing with a special symbol; here we use a lowercase `x`, reminiscent of its common use as a variable.[7] Table 6.5 presents examples of a range of underspecified expressions along with their `L-VAL` attributes using this encoding.

For underspecified expressions referring to times that do not indicate the part of day (either 'am' or 'pm'), such as those in Rows 10, 11 and 14 of the table, we use a lowercase `t` separator (instead of the standard `T` separator) between the date and time parts of the representation. Together, these notational conventions indicate explicitly

[7]Underspecified expressions should not be confused with non-specific expressions; these represent two quite independent semantic phenomena. The former omits some information because it is assumed the reader will be able to retrieve it from the context (e.g. *14th June*), while the latter is typically used generically (e.g. 'The dry season starts in *May*'). In the string-based semantic representation, the underspecified expressions we introduce in LTIMEX use lowercase `x`s (e.g. `xxxx-06-14`), while non-specific expressions, already part of TIMEX2, are annotated with uppercase `X`s (e.g. `XXXX-WXX-7TMO`).

those parts of a temporal value that remain uninstantiated.

The local semantics of bare weekday names, such as *Monday* or *Friday*, cannot be represented in the standard month-based format `yyyy-mm-dd`, and therefore must be represented in the week-based format `yyyy-Wnn-d`, where `nn` is the ISO week number and `d` is the number of the weekday within that week (1 denotes Monday and 7 is used for Sunday).[8]

6.2.3 Offset Expressions

Offset expressions, as we have called them, encode a function which, when applied to a reference time, returns the global semantic value denoting the temporal location of the entity referred to by the expression. This temporal function either adds or subtracts a number of units at some granularity: for example, *last year* is equivalent to subtracting one year from the year of the reference date, and *three days later* means adding three days.

When we introduced our taxonomy of temporal expressions in Chapter 4, we distinguished three subtypes of offset expressions: deictic, anaphoric, and event-based. The difference between them is that for deictic expressions the reference time is the time-stamp of the utterance, for anaphoric expressions the reference time is mentioned somewhere in the context, and for event-based expressions the reference time is the time-stamp of an event mentioned in the discourse. However, given the increased complexity in the case of event-based expressions (the involvement of an event), we discuss their LTIMEX representation in a separate section below, now focusing on the first two subtypes only.

Consider, for example, *tomorrow* and *the following day*; both these expressions refer to a day which comes after the reference date; *tomorrow*, however, can only be interpreted with respect to the date of the utterance (and therefore is deictic), whereas *the following day* cannot be interpreted in this way as this expression is only used in connection with another date mentioned earlier (so it is anaphoric). In LTIMEX, both these expressions have the same offset encoded as the value of the `L-VAL` attribute; the `L-TYPE` attribute indicates whether the expression is `OFFSET-DEICTIC` or `OFFSET-ANAPHORIC`. The interpretation algorithm can use the value of this attribute to decide whether to apply the offset to the time-stamp of the document or to use a temporal focus tracking mechanism to select the correct reference time. If, for any reason, the annotator or a temporal expression tagger cannot determine the subtype of the offset, `L-TYPE` can also be given the value `OFFSET`, leaving the decision about the subtype to the interpretation module.

Table 6.6 presents pairings of deictic and anaphoric date expressions which share the same value for the `L-VAL` attribute. A leading '+' or '−' indicates whether the operation to be performed is addition or subtraction; this is followed by an encoding of the offset that is to be calculated from the reference time. We use an ISO-based format similar to that which TIMEX2 uses for its `VAL` attribute; however, instead of providing a temporal location on a timeline, we put in the individual slots the quantities to be added or subtracted. The position of the slot determines the granularity of the unit: for example, `+0000-00-05` encodes the addition of five days and `-0002` encodes the

[8]As TIMEX2 prefers a calendar date format (also called in the guidelines 'month-based') to express dates, a conversion from the week-based to month-based format must be carried out when deriving the global semantics and representing it in TIMEX2.

No	Deictic Expression	Anaphoric Expression	Representation (`L-VAL`)
1	today	the same day	+0000-00-00
2	tomorrow	the following day	+0000-00-01
3	yesterday	the previous day	−0000-00-01
4	five days ago	five days earlier	−0000-00-05
5	last month	the previous month	−0000-01
6	last summer	the previous summer	−0001-SU
7	two weeks ago	two weeks earlier	−0000-W02
8	(in) two weeks	two weeks later	+0000-W02
9	this weekend	that weekend	+0000-W00-WE
10	this year	that year	+0000
11	three years ago	three years earlier	−0003
12	three years ago today		−0003-00-00
13	next century	the following century	+01

Table 6.6: Examples of the local semantics for offset expressions of dates.

subtraction of two years.[9] Of course, for expressions with zero offset (e.g. *today*) one could use either '+' or '−'; by convention we use '+'.

An offset date expression may be accompanied by unambiguous (e.g. *6 a.m.*) or ambiguous (e.g. *6 o'clock*) information about the time within the referred-to day; see Rows 1–9 of Table 6.7. In these cases only the date component of the expression (e.g. *today* or *tomorrow*) has the form of an offset; here the `T` and `t` separators combine an offset on their left with an absolute value on their right. For example, *6 p.m. two days ago* is represented as `-0000-00-02T18:00`; the leading '−' sign refers here only to the date component, so the encoding does not mean 'subtract 2 days and 18 hours'.

Just as there can be date offsets that have no time information, we can also have time offsets with no date information; for example, *five minutes ago*. In such cases we add the operator ('+' or '−') just after the `T` separator and have a zero date offset. Consider the representation of *eighteen hours and fifteen minutes later*: this has the value `+0000-00-00T+18:15`, making it distinct from the representation of *6:15pm today*, which is `+0000-00-00T18:15`. More examples are provided in Rows 10–15 of Table 6.7.

A time offset may also appear together with a date offset, as shown in Rows 16 and 17 in Table 6.7, or even an explicit or underspecified date, as shown in Rows 18 and 19. In the first case the representation combines a non-zero date offset with a time offset; in the second case we have a non-offset representation of a date followed by the encoding of the time offset. This is consistent with the stance we took in Chapter 4, where we argued that these strings are single temporal expressions.[10]

Finally, we also need to be able to represent offset expressions built on names of weekdays and months, such as *last Monday* or *next March*. In Table 6.8 we present examples with the proper encodings. In our representation we only indicate the direction ($<$ for *last*, $>$ for *next*) and the weekday or month name mentioned in the expression

[9]Note also the difference between the LTIMEX representations of the expressions *three years ago* and *three years ago today*. As the latter expression is referring to a day exactly three years before the reference time, we provide a clue to the granularity by using a zero offset in the relevant slots, i.e. we have `-0003-00-00`; the offset for the former expression is encoded as `-0003`.

[10]Recall that TIMEX2 and TimeML take a different view, in which strings like *8 May 2001, 1 hour later* consist of two temporal expressions.

No	Expression	Representation (`L-VAL`)	Type
1	6 a.m. today	+0000-00-00T06:00	deictic date offset + explicit time
2	6 p.m. that day	+0000-00-00T18:00	anaphoric date offset + explicit time
3	6 p.m. two days ago	−0000-00-02T18:00	deictic date offset + explicit time
4	6 o'clock two days ago	−0000-00-02t06:00	deictic date offset + underspec. time
5	tomorrow morning	+0000-00-01TMO	deictic date offset + explicit time
6	morning the day before	−0000-00-01TMO	anaphoric date offset + explicit time
7	last night	−0000-00-01TNI	deictic date offset + explicit time
8	11pm last night	−0000-00-01T23:00	deictic date offset + explicit time
9	2am last night	+0000-00-00T02:00	deictic date offset + explicit time
10	two hours earlier	+0000-00-00T−02	anaphoric time offset
11	an hour and 20min later	+0000-00-00T+01:20	anaphoric time offset
12	(in) six hours time	+0000-00-00T+06	deictic time offset
13	five minutes ago	+0000-00-00T−00:05	deictic time offset
14	(in) sixty seconds	+0000-00-00T+00:00:60	deictic time offset
15	sixty seconds later	+0000-00-00T+00:00:60	anaphoric time offset
16	tomorrow 2 hours later	+0000-00-01T+02	deictic date offset + time offset
17	the next day 2hrs later	+0000-00-01T+02	anaphoric date offset + time offset
18	8 May 2001, 1 hour later	2001-05-08T+01	explicit date + time offset
19	17 May, one hour earlier	xxxx-05-17T-01	underspecified date + time offset

Table 6.7: Examples of the local semantics for offset expressions with references to times of day.

(`Dn` and `Mnn`, respectively). It will be the task of the interpretation stage to determine which calendar week and year is intended. Expressions using the determiner *this* (e.g. *this Wednesday* or *this June*) are treated as underspecified expressions unless the determiner is used together with other tokens indicating the direction of interpretation (e.g. *this coming Wednesday*, for which the encoding is >`D3`).

6.2.4 Event-based Point Expressions

An event-based expression identifies a temporal entity by means of a reference to an event. In such expressions, the `L-TYPE` attribute has the value `L-TYPE=EVENT-BASED-POINT`, and we provide the identifier of the event in the `L-EVENT_ID` attribute. If an application does not perform event-recognition, or in a given circumstance is unable to identify the event in question, then the value of this attribute is left empty.

In some cases the temporal value is expressed as an offset to the time of an event, as in Example (6.7):

(6.7) a. *Ten seconds after the second explosion* the plane hit the ground.
`L-VAL=+0000-00-00T+00:10  L-TYPE=EVENT-BASED-POINT  L-EVENT_ID=`e

b. Jane got a salary raise *the day after Michael lost his job.*
`L-VAL=+0000-00-01  L-TYPE=EVENT-BASED-POINT  L-EVENT_ID=`e

c. Jane got a salary raise *the day that Michael lost his job.*
`L-VAL=+0000-00-00  L-TYPE=EVENT-BASED-POINT  L-EVENT_ID=`e

Here the `L-VAL` attribute encodes the offset, just as it does in offset point expressions. The specified type of the expression indicates that the reference time to be used in the

No	Deictic Expression	Anaphoric Expression	Representation (`L-VAL`)
1	last Monday	the previous Monday	<D1
2	next Wednesday	the next Wednesday	>D3
3	this coming Wednesday	that coming Wednesday	>D3
4	this Wednesday	that Wednesday	xxxx-Wxx-3 or D3
5	last June	the previous June	<M06
6	next June	the next June	>M06
7	this June	that June	xxxx-06

Table 6.8: Examples of the local semantics for offset expressions containing the name of a month or a weekday.

interpretation is the time of the event indicated by the *e* event variable.

In cases where the time denoted by the expression can be computed from the time-stamp of the event simply by refining its granularity, we use a zero offset just to indicate the granularity (temporal unit) of the result. Consider the following example:

(6.8) I met my wife *the year when I bought my house.*
`L-VAL=+0000  L-TYPE=EVENT-BASED-POINT  L-EVENT_ID=`e

The temporal value of the expression is to be calculated here by adding zero years to the year of the event time-stamp, and discarding any more detailed information that the time-stamp might provide (e.g. the month and day).

In other cases, the time denoted by the expression may be exactly the time of the event, as in the following example:

(6.9) At *the time of the peace agreement* the United States agreed to replace equipment on a one-by-one basis.
`L-VAL=EVENT_TIME  L-TYPE=EVENT-BASED-POINT  L-EVENT_ID=`e

Note that the expression does not indicate the granularity. In such cases, the `L-VAL` attribute contains the `EVENT_TIME` token, which means that the temporal value is the time of the underlying event.

For point temporal expressions which refer to a part of an event, as in Example (6.10), we use the encoding of ordinally-specified expressions, which we discuss in detail in Section 6.2.8:

(6.10) The casualties included 19,240 dead on *the third day of the Battle of the Somme.*
`L-VAL=3D  L-TYPE=EVENT-BASED-POINT  L-EVENT_ID=`e

The `3D` value tells us that, of the whole time span of the event, the expression refers only to the third day. We specify the type of the expression to be `EVENT-BASED-POINT` because a day refers to a calendar date, which in our framework is a point on a timeline. With other temporal units, their alignment with points on a timeline is not obvious. Consider the following example:

(6.11) We did most work in *the first month of the project.*
`L-VAL=1M  L-TYPE=EVENT-BASED  L-EVENT_ID=`e

Here, the month mentioned in the example may start, for example, sometime around the middle of a calendar month; in such a case it is treated as a period. But if the project started at a beginning of a month, for example 1 June, then *the first month* would be a point, a calendar month. Because we can not resolve this ambiguity at the level of local semantics, we specify the type of the expression as `EVENT-BASED`, leaving the decision whether to convert the ordinally-specified value `1M` into a point (a specific calendar month) or a period of one month to the interpretation algorithm.

6.2.5 Period Expressions

For expressions that denote periods, `L-VAL` takes the same values as the corresponding `VAL` attribute in TIMEX2; this also covers those cases where the duration mixes different units, as in the following example:

(6.12) This project will run for *one year and two months.* `L-VAL=P1Y2M`

The anchoring attributes are to be filled in only if the anchor is mentioned within the extent of the expression. The anchor may be provided in various forms: it may be explicit (see Example (6.13a)), underspecified (see Example (6.13b)) or offset (see Example (6.13c) and (6.13d)). In each case, the `L-ANCHOR_VAL` attributes encode that anchoring point in one of the formats we have already introduced:

(6.13) a. The accounts are paid in full for *the six months ended 31 March 2009.*
`L-VAL=P6M L-ANCHOR_VAL=2009-03-31 L-ANCHOR_DIR=ENDING`

b. The accounts are paid in full for *the six months ended March 31.*
`L-VAL=P6M L-ANCHOR_VAL=xxxx-03-31 L-ANCHOR_DIR=ENDING`

c. The renovations will last *five days starting tomorrow.*
`L-VAL=P5D L-ANCHOR_VAL=+0000-00-01 L-ANCHOR_DIR=STARTING L-ANCHOR_TYPE=DEICTIC`

d. The movie festival will end on *18 July*, but then we have the theatre workshops that will run for *a whole week starting just the very next day.*
`L-VAL=P1W L-ANCHOR_VAL=+0000-00-01 L-ANCHOR_DIR=STARTING L-ANCHOR_TYPE=ANAPHORIC`

In the last example above we also use the `L-ANCHOR_TYPE` attribute to encode the type of the offset of the anchor; the possible values here are `DEICTIC` and `ANAPHORIC`. The expression may be also anchored implicitly, as in the following example:

(6.14) *The next three days* were extremely hot and humid.
`L-VAL=P3D L-ANCHOR_VAL=+0000-00-00 L-ANCHOR_DIR=STARTING`

In such cases we provide the offset in the `L-ANCHOR_VAL` attribute, but leave it to the interpretation algorithm to decide (for example, based on the tense of the sentence) whether the anchor is deictic or anaphoric.

If the expression itself does not state when the period starts or ends, then no anchor-related attributes are specified:

(6.15) The Nile Movie Festival lasted *five days.* `L-VAL=P5D`

If the rest of the document provides such information, the anchor is to be determined in the interpretation stage, when the global semantics is derived; this also means that the final annotation does not have the `L-ANCHOR_VAL` and `L-ANCHOR_DIR` attributes, it only has `ANCHOR_VAL` and `ANCHOR_DIR`.

6.2.6 Event-based Period Expressions

For event-based periods, we encode the duration in the `L-VAL` attribute just as in the case of the durations discussed in Section 6.2.5, but the type of the expression in the `L-TYPE` attribute is specified as `EVENT-BASED-PERIOD`. Similarly to the annotation of event-based point expressions, we provide the identifier of the underlying event in the `L-EVENT_ID` attribute. The time of the event, however, does not serve here as the reference time to be used in the following interpretation stage to calculate the value of the `VAL` attribute; rather, it determines the location of the period, and is used to compute one of the period's anchors. Consider the following examples:

(6.16) a. The rate of US combat deaths in Baghdad nearly doubled in *the first seven weeks of the "surge" in security activity.*
`L-VAL=P7W L-ANCHOR_VAL=EVENT_START L-ANCHOR_DIR=STARTING`
`L-EVENT_ID=`*e* `L-TYPE=EVENT-BASED-PERIOD`

b. *The last three days of the battle* were extremely brutal.
`L-VAL=P3D L-ANCHOR_VAL=EVENT_END L-ANCHOR_DIR=ENDING`
`L-EVENT_ID=`*e* `L-TYPE=EVENT-BASED-PERIOD`

c. I was so panicked I could not take a single step for *30 minutes after the earthquake.*
`L-VAL=PT30M L-ANCHOR_VAL=EVENT_END L-ANCHOR_DIR=STARTING`
`L-EVENT_ID=`*e* `L-TYPE=EVENT-BASED-PERIOD`

d. There was no terrorist warning in *the three years before the bombing in the underground.*
`L-VAL=P3Y L-ANCHOR_VAL=EVENT_START L-ANCHOR_DIR=ENDING`
`L-EVENT_ID=`*e* `L-TYPE=EVENT-BASED-PERIOD`

To handle the fact that events themselves span over some periods of time, we introduce two tokens, `EVENT_START` and `EVENT_END`, to be used in the `L-ANCHOR_VAL` attribute. These tokens indicate which end of the period of the event is to be used as the anchor.

6.2.7 Modified Expressions

The `L-MOD` attribute augments the semantic value encoded in the `L-VAL` attribute by specifying a modifier. The modifier may shift the temporal location of an entity (see Example (6.17a)), or focus on a part of a temporal unit (see Example (6.17b)), or change the duration of a period (see Example (6.17c)).

(6.17)	a.	Anne took her final exams *more than a month ago.*	`L-MOD=BEFORE`
	b.	John visited us in *mid-June.*	`L-MOD=MID`
	c.	Painting this fence will take *at least one week.*	`L-MOD=EQUAL_OR_MORE`

In our framework, the `L-MOD` attribute takes the same values as the `MOD` attribute in TIMEX2, and these are used in the same manner.

6.2.8 Ordinally-specified expressions

Some temporal expressions use what we call ordinally-specified elements; for example, the expressions in Examples (6.18a)–(6.18c) make reference to a specific day by means of selecting the third day of some coarser temporal unit or an event.

No	Expression	Representation (`L-VAL`)
1	the first Tuesday	1D2
2	the third day	3D
3	the last Tuesday	$1D2
4	the last day	$1D
5	the last but one day	$2D
6	the last month	$1M
7	the last February	$1M02

Table 6.9: Examples of the local semantics for expressions with ordinally-specified references.

(6.18) a. *the third day of next month*

b. *the third day of the last decade*

c. *the third day of the trip*

To encode such ordinally-specified elements we use the format `$nu`, where `n` is a number, `u` indicates the temporal unit to be used, and `$` is an optional marker used when the ordinal is to be counted from the end of some chunk of time (e.g. *last*, *penultimate*). Examples of ordinally-specified elements of expressions and their representations are shown in Table 6.9.

The expressions using ordinally-specified elements are annotated with multiple TIMEX2 annotations, as shown in Examples (6.19)–(6.21). The ordinally-specified format is recorded only in the outermost annotation; the inner expression, which may be, for example, underspecified or an offset, receives its own proper representation of its local semantics.

(6.19) <TIMEX2 L-VAL="3D">*the third day of*
<TIMEX2 L-VAL="+0000-01">*the next month*</TIMEX2></TIMEX2>

(6.20) <TIMEX2 L-VAL="$1D1">*the last Monday of*
<TIMEX2 L-VAL="xxxx-05">*May*</TIMEX2></TIMEX2>

(6.21) <TIMEX2 L-VAL="1D">*the first day of*
<TIMEX2 L-VAL="2M">*the second month of*
<TIMEX2 L-VAL="+0001">*next year*</TIMEX2></TIMEX2></TIMEX2>

When deriving the global semantics from the local semantics, the individual values of the nested expressions must be combined together; the process is carried out recursively from the outermost to the innermost, resolving the temporal references while backtracking from the innermost to the outermost.

The type recorded in `L-TYPE` of an expression whose `L-VAL` is ordinally-specified is the same as the type of its innermost expression; Example (6.19) is an anaphoric offset, Example (6.20) is underspecified, and Example (6.21) is deictic.

6.2.9 Non-Specific Expressions

In many cases the decision that a temporal expression is non-specific can be only made in the course of analysing the sentence that contains the expression; and sometimes an even wider context may need to be considered.

For example, consider the generic references to months in the following sentence:

(6.22) In the southern hemisphere *days* are much longer in *January* than in *July*.

These are not obviously non-specific when we consider only the extent of the expressions themselves. The local semantic representations are therefore underspecified, i.e. `xxxx-01` and `xxxx-07`. In the interpretation stage, the lowercase `x`s must not be instantiated with a specific year, but must be converted into markers of non-specificity (uppercase `X`s, for example, if TIMEX2 is the scheme used for global semantics representation).

Similarly, it could at first seem that indefinite point temporal expressions, which can only take the form of indefinite noun phrases, can be recognized as non-specific already at the level of local semantics:

(6.23)	a.	I was born on *a Sunday*.	`L-VAL=xxxx-Wxx-7`	`VAL=XXXX-WXX-7`
	b.	I met my wife on *a sunny day in July*.	`L-VAL=xxxx-07-xx`	`VAL=XXXX-07-XX`

However, the following example shows that indefinite noun phrases can also be used to refer to specific times:

(6.24)	This year, 15th July is *a Saturday*.	`L-VAL=xxxx-Wxx-6`	`VAL=2004-07-15`

In the above sentence, the expression *a Saturday* refers to a specific date that in the interpretation stage must be resolved to the 15th July of a year that needs to be determined deictically.[11]

Periods of indefinite duration, such as *a few days*, can also be recognized as non-specific without reference to the context. The encoding of such durations uses `X` instead of a specific number, e.g. `PXD`.

Similarly, some set expressions can be identified as non-specific already at the stage of local semantic analysis; for example, *every few days* or *some Mondays in 2004*. Unfortunately, TIMEX2 is unable to represent the semantics of these expressions correctly,[12] and in consequence our representation fails here too.

6.2.10 Set Expressions

The semantic representation of set expressions is complex, because these expressions do not refer to a single entity, but to a set of entities. Neither TIMEX2 nor TimeML express the semantics of these expressions sufficiently well to make these schemes applicable to all set expressions. As an alternative, Pan's (2007) first-order logic representation for set expressions, which is formally sound and has much broader coverage, can be encoded in OWL; but the complexity of OWL goes far beyond the goals of TIMEX2 and TimeML.

As indicated earlier, our aim is to provide a representation for local semantics that is compatible with the use of TIMEX2 for representing the global semantics of temporal expressions. Inevitably, this compromises any attempts to appropriately represent sets at the level of local semantics.

We indicate the set type by assigning the `YES` value to the `L-SET` attribute (following the use of the `SET` attribute in TIMEX2), and we attempt to specify any underspecification or offset that might appear, as in the following examples:

[11]The value of the `VAL` attribute provided in the example assumes the sentence was uttered in year 2004.

[12]For example, *some Mondays in 2004* is represented just in the same way as *all Mondays in 2004*: `VAL=2004-WXX-1, SET=YES`.

(6.25) a. *every winter in the 80s* `L-VAL=xx8-WI L-SET=YES`

b. *monthly* `L-VAL=xxxx-XX L-SET=YES`

In some other cases we may be able to obtain a reliable representation by tweaking the use of attributes and their possible values in TIMEX2; for instance, in Example (6.26a) the expression is represented by means of any period of two years with its ending anchored on years having zero as their final digit (e.g. 1960, 1990, 2000). In Example (6.26b) we do something similar, but we anchor the periods on the last day of a month (and in doing so we specify the anchor with the format used for ordinally-specified references).

(6.26) a. *the last two years of every decade*
`L-VAL=P2Y L-SET=YES`
`L-ANCHOR_VAL=XXX0 L-ANCHOR_DIR=ENDING`

b. *the last two days of every month*
`L-VAL=P2D L-SET=YES`
`L-ANCHOR_VAL=XXXX-XX-$1D L-ANCHOR_DIR=ENDING`

c. *the last two days of every month next year*
`L-VAL=P2D L-SET=YES`
`L-ANCHOR_VAL=+0001-XX-$1D L-ANCHOR_DIR=ENDING`

This, however, already goes beyond the TIMEX2 rules, which prohibit using the anchor attributes for set expressions (Ferro et al., 2005, p. 42). Note also that `$1D`, the ordinally-specified element in Examples (6.26b and c), cannot be converted to any specific number during interpretation because the month is non-specific and months have different durations; but the value `$1D` is not supported in TIMEX2.

More complex expressions, such as *every Tuesday and Friday until the end of the year* are problematic to represent using TIMEX2 and therefore using LTIMEX. In TIMEX2 such a string is treated as three distinct expressions annotated with no overlap in their extent or referential links that would indicate the relationship between them:

(6.27) *every Tuesday* and *Friday* until *the end of the year*

With such a decomposition, the local semantics is encoded as follows:

(6.28)

every Tuesday	`L-VAL=xxxx-WXX-2`	`L-SET=YES`	
Friday	`L-VAL=xxxx-Wxx-5`	`L-SET=NO`	
the end of the year	`L-VAL=xxxx`	`L-SET=NO`	`L-MOD=END`

Just as in the representation of the global semantics, the local semantics here is far from representing the real meaning of the expression. If we wanted to represent the whole string from Example (6.27) as a single temporal expression, the semantic representation would need to be much richer to specify all the temporal constraints (e.g. the conjunction of Tuesdays and Fridays) that let us identify which entities belong to the set.

6.2.11 Summary

We have developed a string-based representation of the local (i.e. context-independent) semantics of temporal expressions, which we called LTIMEX. It can be easily combined with the existing annotation schemes (specifically, TIMEX2 and TimeML) which currently allow only for the representation of the global (i.e. fully-interpreted) semantics.

We are thus proposing an extension to these schemes that provides a means of support for an additional level of semantic representation; this in turn supports a modular design for temporal expression tagging, with a well-defined interface between the recognition and interpretation modules, and consequently allows for a more detailed evaluation of taggers.

Table 6.2 summarises the attributes used in LTIMEX and their values. We use in total eight attributes: three are used in the same way as their TIMEX2 counterparts (`L-MOD`, `L-SET` and `L-ANCHOR_DIR`); `L-VAL` represents the partial meaning of the expression;[13] similarly, `L-ANCHOR_VAL` encodes information about the anchor of a period; and three attributes are completely new: `L-TYPE`, which encodes the taxonomical type of the expression; `L-EVENT_ID`, which for event-based expressions stores the identifier of the event; and `L-ANCHOR_TYPE`, which, for durations with the anchor expressed by means of an offset, encodes whether it is deictic or anaphoric.

An area left for future work is the improvement of the representation of set expressions. This could perhaps be aligned with further development of TimeML, which is to become an ISO standard (see the discussion in (Pustejovsky et al., 2010)); although this takes a step further compared to TIMEX2, it still does not have a proper means to represent the global semantics of set expressions.

6.3 Temporal Focus Tracking

In this section we investigate the problem of **temporal focus tracking**: finding in the text a temporal expression whose value (the reference time) is used to interpret a context-dependent temporal expression. We present the phenomena related to temporal focus tracking and indicate the challenges any algorithm must address. We further review existing work, suggest possible solutions and present the results of our experiments evaluating their performance. We close the section with conclusions drawn from the experiments.

6.3.1 The Phenomenon

In the case of fully-specified (explicit) expressions, the values of local and global semantics are equal, because their meaning does not depend on the context. For example, in a given calendar, the expression *April 9, 1865* always means the ninth day of the fourth month of year 1865. Things are not so simple in the case of context-dependent expressions: unlike an explicit expression, a context-dependent expression may refer to a completely different entity if used in a different context, despite there being no change in the content of the expression itself. Consider the offset expression *the next day*: this appears twice in Figure 6.2,[14] once in the second paragraph and then in the third paragraph. Independently of where the expression is used, its local semantics in LTIMEX is encoded uniformly as `L-VAL=+0000-00-01`. However, the first occurrence refers to the date 1865-04-09, and the second occurrence refers to 1865-05-11.

[13]It is partial in the sense that it does not capture information about temporal modifiers and anchors, which are encompassed in separate attributes: `L-MOD`, `L-ANCHOR_DIR`, and `L-ANCHOR_VAL`.

[14]Figure 6.2 presents a fragment of the article about the American Civil War sourced from Wikipedia; see `http://en.wikipedia.org/w/index.php?title=American_Civil_War&oldid=346626531`.

Lee surrendered his Army of Northern Virginia on *April 9, 1865*, at the McLean House in the village of Appomattox Court House. In an untraditional gesture and as a sign of Grant's respect and anticipation of peacefully folding the Confederacy back into the Union, Lee was permitted to keep his officer's saber and his horse, Traveller. On *April 14, 1865*, President Lincoln was shot. Lincoln died *early the next morning*$_{[1865\text{-}04\text{-}15]}$, and Andrew Johnson became President.

Events leading to Lee's surrender began with the capture of key Confederate officers Richard S. Ewell and Richard H. Anderson on *April 6*$_{[1865\text{-}04\text{-}06]}$, following Confederate defeat at the battle of Sayler's Creek. On *April 8*$_{[1865\text{-}04\text{-}08]}$ Union cavalry under Major General George Armstrong Custer destroyed three trains of Confederate supplies at Appomattox Station, leading to the surrender of General Lee *the next day*$_{[1865\text{-}04\text{-}09]}$. General St. John Richardson Liddell's army surrendered after the loss of the Confederate fortifications at the Battle of Spanish Fort in Alabama, also on *April 9*.

On *April 21* John S. Mosbys raiders of the 43rd Battalion Virginia Cavalry disbanded and on *April 26* General Joseph E. Johnston surrendered his troops to Sherman at Bennett Place in Durham, North Carolina. Surrendering *on May 4* and *5* were the Confederate departments of Alabama, Mississippi and East Louisiana regiments and the District of the Gulf. The Confederate President was captured on *May 10* and the surrender of the Department of Florida and South Georgia happened *the same day*. Confederate Brigadier General "Jeff" Meriwether Thompson surrendered his brigade *the next day*$_{[1865\text{-}05\text{-}11]}$ and *the day following*$_{[1865\text{-}05\text{-}12]}$ saw the surrender of the Confederate forces of North Georgia.

Figure 6.2: A fragment of a Wikipedia article about the American Civil War.

The text in Figure 6.2 also contains a number of underspecified expressions; consider, for example, *April 6* and *April 9* as they appear in the second paragraph. These expressions do not carry information about the year component of the dates, and without knowing the context we do not know what dates are actually meant. A human reader can link them with a proper temporal expression appearing earlier in the text to figure out that these dates both refer to days in 1865. The goal of the interpretation stage, and in particular of a temporal focus tracking mechanism, is to infer this information by finding the proper reference expression.

The expression serving as the reference time for the interpretation of another expression does not need to be fully-specified. For example, in the third paragraph in Figure 6.2 we find the following chain of dependencies between the expressions:

(6.29) *May 10* → *the same day* → *the next day* → *the day following*

In the domain of news stories, interpreting temporal expressions by using the publication time-stamp as the reference time generally gives the correct results; see, for example, the work of Mani and Wilson (2000b). However, such an approach is obviously very simplistic and does not work for many documents from outside of this domain. The publication time-stamp (or any form of a document creation date) only works for deictic expressions, which in some domains—such as historical narrative—may be very uncommon.

In some cases the reference time for a temporal expression may be the time of an event. Consider the following example:

(6.30) The fort surrendered when more British troops arrived. *Two days later*, the war was over.

Here, in order to interpret the expression *two days later* we need to determine the time-stamp of the event *the fort surrendered.* The processing of such event references is beyond the scope of the work presented here.

A fragment of quoted text should be treated as a subdocument of the original text with its own time-stamp, since the quote presents an utterance that may have been produced at a different time than the time of the quoting text. In Example (6.31), the expressions *today*, *tomorrow* and *into the weekend* are deictic expressions for which the expression *late Thursday* is the utterance time.

(6.31) "We will be examining them *today*, *tomorrow* and *into the weekend*," he said *late Thursday.*

Depending on the genre of documents, these cases may occur more or less frequently.

6.3.2 Related Work

The notion of temporal focus was introduced by Nakhimovsky (1987); his work arose from the even broader notion of 'focus of attention' introduced by Grosz (1977) in order to manage references to the objects and actions mentioned in a discourse. Their work set up foundations for solutions that might be used in a temporal expression tagger—for example, the stack model which we discuss in more detail below—but is of a rather more theoretical nature than we deal with and is also broader in scope than our work by, for example, considering time progression expressed using tense and aspect. Our focus is on implementable algorithms and heuristics capable of resolving a reference time provided in the text by a temporal expression for the interpretation of another temporal expression which is context-dependent.[15]

The existing literature on determining the reference time in temporal expression taggers comes from two periods. The earlier period—the second half of 1990s—delivered a number of systems capable of processing dialogs in which participants discussed the scheduling of meetings. In some cases these systems were large-scale commercial solutions, which unfortunately limits the amount of published detail. The second period started at the ACE 2004 evaluations, and began a new wave of interest, which is still ongoing, in research on the processing of temporal expressions. However, it seems that not much attention has been paid to improving temporal focus tracking techniques. In both these periods the dominant method was some variation of a recency model—i.e., using the most recent temporal expression as the temporal focus—but there may be various constraints added concerning the type and granularity of the reference time expression.

Overall, we have found references to four major solutions and models: using the utterance time, the recency model, the stack model and the graph-based model. The last of these appears not to have been implemented.

[15]Similarly, the concerns we address are generally quite distinct from those discussed in the literature on 'temporal anaphora', a term introduced by Partee (1973), who argued that time adverbials play the same role in the tense system as phrases which serve as the antecedents of pronouns. This idea was further developed, for example, by Partee (1984), Hinrichs (1986) and Moens and Steedman (1987). For a more complete bibliography on this topic see `http://www.utsc.utoronto.ca/~binnick/old%20tense/tempAna.html`.

6.3.2.1 Using the Utterance Time

The simplest technique is to use the time of the utterance as the reference time for every context-dependent expression. Berglund (2004) claimed satisfactory results with such an approach, but did not present precise numbers. We suspect that in his domain of police reports on car accidents the majority of the expressions were deictic, which by definition are supposed to be interpreted with respect to the time-stamp of the utterance.

For Bittar (2009), the utterance-time method turned out to be very unreliable, but also in this case no precise results were reported. In his interpretation system, a candidate value for a temporal expression was obtained by using the document time-stamp as a reference, but had to be confirmed and, if necessary, corrected by the user.

Mani and Wilson (2000b) noted that of all 94 errors concerning the `VAL` attribute of correctly recognized expressions, only 14, which corresponds to 14.9%, were due to the incorrect reference time being used.[16] Since the system generated in total 680 values of the attribute, the erroneous reference time constitutes a 2.1% error rate. Similarly, Wu et al. (2005b) reported that for their tagger for Chinese, in 11.3% of the cases when the `VAL` attribute was incorrect (122 out of 1083 values), the reason was the incorrect selection of the reference time; this corresponds to 2.8% of all 4290 temporal expressions found in their gold-standard dataset. However, in both cases the authors did not check whether fixing other sources of errors (e.g. wrong extent recognition, or providing an empty value) would change the ratios. In consequence this error rate cannot be considered a precise evaluation of this method of addressing the problem of temporal focus tracking.

6.3.2.2 The Linear-recency Model

The recency-based model (see, for example, Hobbs (1978)) has proven popular, no doubt due to its relatively good performance (reported for example by Wiebe et al. (1998) and Han et al. (2006a)) and ease of implementation. As the name suggests, in this model the reference time is selected from one of the temporal expressions occurring recently before the expression being interpreted. Most often, the expression used is the most recent one; usually additional conditions are imposed, such as the requirement that the expression is of the required taxonomical type (e.g. explicit) or of appropriate granularity (e.g. the same as the expression being interpreted, one level coarser, one level finer, any level finer).

Wiebe et al. (1998) carried out an empirical study on two parallel Spanish–English corpora referred to as the CMU dialogs and the NMSU dialogs.[17] Both corpora contain transcripts of discussions concerning the scheduling of meetings; however, the participants of the NMSU dialogs also make some comments about their life commitments and discuss topics not relevant to the task. The goal of the study was to check the usefulness of the recency model for this particular domain. In total there were 292 temporal expressions in the two corpora: 196 in CMU and 96 in NMSU. The data

[16]The 94 errors split into: 39 missing values, 25 spurious values (these were all generic expressions which did not receive any value in the gold standard), and 30 expressions with an incorrect value (14 of which were due to the use of utterance time as the reference time).

[17]The CMU corpus was created under the JANUS project, which later became a foundation for the TIDES Parallel Corpus (see Section 2.5.2). The NMSU corpus was collected at New Mexico State University, hence its name.

analysis showed that for 229 out of 238 (96.22%) of anaphoric relations across both datasets (98.2% for CMU and 91.6% for NMSU), the correct reference date was the immediately previous temporal expression, in five cases the second last temporal expression yielded the correct interpretation, and in the remaining four cases one had to use the third-most recent temporal expression.

Han et al. (2006b) also constructed a system applied to the domain of meeting scheduling; however, they processed email threads rather than speech transcripts. They restricted the recency model with the condition that a noun-modifying expression could not function as a temporal focus. This was motivated by cases like that in Example (6.32), where *Sunday* is not related in any way to the expression *2005*.[18]

(6.32) We received a copy of *2005* report and will send you our analysis by *Sunday*.

Their development and training data contained in total 781 emails with 1136 temporal expressions (107 explicit, 433 deictic, 545 anaphoric offsets and underspecified, 52 durations). Given the large proportion of explicit and deictic expressions their baseline method was to use the document's time-stamp; this resulted in 50% accuracy (measured on all temporal expressions).[19] The accuracy results for the module performing temporal focus tracking are 78.2%, 85.45% and 76.34% on two development and one test datasets, respectively.[20] One of the ACE 2007 participants used Han et al's (2006) TCNL representation framework but modified the interpretation mechanism by resetting the temporal focus to the document time-stamp after each paragraph; the influence of this modification on the results was not reported.

In Chronos, a system developed by Negri and Marseglia (2005) and submitted to the ACE 2004 evaluation, the temporal focus was selected as the nearest previous expression of the same or finer granularity. So, for example, *July 2008* could be used as a reference date for the expressions *next month* or *next year*, but it would not be selected to interpret the expressions *tomorrow* or *in two weeks*. We note that such an approach would not work for some underspecified expressions, such as *February* in Example (6.33).

(6.33) *Year 2005* was closed with a negative balance resulting in some dismissals in *December*. The bad luck started for the company already in *January*. But we had lots of success in *2006*. We obtained 10 new customers already in *February*.

This is because while an offset requires the reference date to be at least as fine-grained as the expression in question, an underspecified expression requires the reference that provides the missing date/time information. The authors do not explain what Chronos did for underspecified expressions; we may guess that the most recent expression that could fill the missing slots was used. Unfortunately, there is no direct evaluation of the recency model used and its impact on the performance of the system. The classification of deictic and anaphoric expressions was carried out with a few simple rules: *yesterday*, *today*, *tonight*, month names, weekday names, and any expression with the keywords

[18]We note that the sentence in Example (6.32), presented here in the original version sourced from (Han et al., 2006b), is not grammatical, since a determiner is missing in front of the temporal expression *2005*. However, this grammatical error does not impact the issue raised by the authors.

[19]In the evaluation the extents of all expressions were already provided on input, so they report accuracy instead of precision and recall.

[20]However, the improvement was not only due to the recency model, but also thanks to the incorporation of tense and aspect information in the interpretation stage, and several representational improvements; additionally the errors made concern not only incorrect selection of reference expressions, but also other sources of errors.

this, *last*, *next*, *past* or *ago* were considered deictics, and expressions with any of the keywords *following*, *previous*, *same*, *that*, *before*, or *later* were considered anaphoric.

Kimura et al. (2007) used the most recent ('adjacent') temporal expression; although this is not stated explicitly in their paper, we assume only expressions of appropriate granularities were considered. The system dealt only with dates of year, month or day granularity because it was created to process biographical data of famous people (in Japanese). Out of the 1159 context-dependent expressions sourced from 500 web pages, 834 were selected for evaluation (325 expressions were discarded from the evaluations because the first date occurring in text did not specify the year, thus making it impossible to fully interpret some expressions). The system correctly interpreted 71.2% of the expressions, with the following precision for particular types: 74.3% (underspecified), 62.3% (coreference—zero offset from the reference time) and 42.6% (non-zero offset). The main reason for the poor performance when interpreting offset expressions was that many of these were deictic.

6.3.2.3 The Stack Model

The earliest implementations of the stack model were two systems co-developed by the German Research Center for Artificial Intelligence (DFKI): Cosma (Busemann et al., 1994) and Verbmobil (Alexandersson et al., 1997). The former provided a German natural language interface (see (Busemann et al., 1997)) for an appointment scheduling agent, and the latter was a speech-to-speech translation system designed to handle dialogues about the scheduling of meetings. In Cosma, the temporal focus for an expression was first looked for in the previous part of the speaker's turn in the dialogue, with the most recent temporal expression being chosen. If none was found, then the dialogue history was searched, where this was organised as a stack based on the structure of the preceding discourse.[21] If this search also failed, then the expression was treated as deictic, i.e. it was interpreted with respect to the time-stamp of the utterance. We could not find any details of the approach taken in Verbmobil, and there do not appear to be any reported evaluation results for the modules used in either system for tracking the temporal focus.

6.3.2.4 The Graph-based Model

Rose et al. (1995) proposed a graph-based discourse structure representation as an alternative to the stack model, in order to represent dialogs with multiple threads, where multiple propositions are negotiated in parallel. Their examples of such dialogs are presented in Figure 6.3, where two candidate days for a meeting are discussed (Tuesday and Wednesday), and in Figure 6.4, where speakers first discuss possibilities of a meeting in one week and then one week later. However, no evaluation of using such a representation for the interpretation of temporal expressions has been published. It is also not clear how common such structures really are: Wiebe et al. (1998) did not find in their corpora a single dialog with such phenomena.

[21]The stack model for discourse was proposed by Grosz and Sidner (1986): in this model, discourses are considered to be hierarchically structured, and at any point in time the stack contains spaces corresponding to currently open discourse segments, sometimes referred to as the right frontier of the tree. When a discourse segment ends, the corresponding space is popped off the stack.

(S1) When can you meet *next week*?

(S2) *Tuesday afternoon* looks good. I could do it *Wednesday morning* too.

(S1) *Tuesday* I have a class *from 12:00-1:30*. But *the other day* sounds good.

Figure 6.3: An example of a multithreaded dialog between two speakers scheduling a meeting and considering different days within the same week.

(S1) We need to set up a schedule for the meeting. How does your schedule look for *next week*?

(S2) Well, *Monday* and *Tuesday both mornings* are good. *Wednesday afternoon* is good also.

(S1) It looks like it will have to be *Thursday* then. Or *Friday* would also possibly work. Do you have time between *twelve* and *two* on *Thursday*? Or do you think sometime *Friday afternoon* you could meet?

(S2) No. *Thursday* I have a class. And *Friday* is really tight for me. How is *the next week*? If all else fails there is always video conferencing.

(S1) *Monday*, *Tuesday*, and *Wednesday* I am out of town. But *Thursday* and *Friday* are both good. How about *Thursday* at *twelve*?

(S2) Sounds good. See you then.

Figure 6.4: An example of a multithreaded dialog between two speakers deliberating over a meeting time in a two-week time frame.

6.3.2.5 Summary

Using the utterance time as the temporal focus is the easiest approach to implement, and, for those domains and text genres where there is a high proportion of deictic expressions, gives satisfactory results. However, for non-deictic expressions this approach is less likely to work. Since deictic expressions are a well-defined category, these could be even excluded from the evaluation of temporal focus tracking algorithms, so that only expressions involving underspecified and anaphoric offsets are considered. A temporal expression tagger must then, however, correctly classify whether an expression is deictic or anaphoric. This can be done either with hand-crafted rules or machine-learned classifiers.

For the interpretation of anaphoric expressions the recency model seems to be the most common approach reported in the literature. However, the vast majority of system descriptions reporting the application of this method do not present a direct evaluation of it, and the ones that do are still quite superficial and do not analyse the problem in detail. As the domains for which this model was used and reported on in the literature are mainly news and dialogues about meeting scheduling, it is not clear how well this model functions outside of these domains. Moreover, different authors use additional conditions concerning the selection of the reference time, which makes their work incomparable. Also, with the exception of Kimura et al. (2007), no evaluations have considered underspecified expressions and anaphoric offsets separately.

The stack and graph models are very difficult to implement as they require a sophisticated analysis of discourse structure; and as we noted, Wiebe et al. (1998) suggest that the latter may be required very rarely in any case.

6.3.3 The Experimental Set-up

6.3.3.1 The Evaluation Methodology

As we noted above, the goal of the temporal focus tracking is to find the reference time that can be used to interpret a context-dependent temporal expression. For example, the expression *the following year* in Example (6.34), in order to be interpreted, must be linked to an expression occurring earlier in the text so that we know that the offset must be calculated from year 492 BC.

(6.34) The first campaign, in *492 BC*$_{\text{[ID=t32, val=BC0492]}}$, was led by Darius's son-in-law Mardonius, who re-subjugated Thrace, which had nominally been part of the Persian empire since *513 BC*$_{\text{[ID=t33, val=BC0513]}}$. Mardonius was also able to force Macedon to become a client kingdom of Persia; it had previously been allied but independent. However, further progress in this campaign was prevented when Mardonius's fleet was wrecked in a storm off the coast of Mount Athos. Mardonius himself was then injured in a raid on his camp by a Thracian tribe, and after this he returned with the remainder of the expedition to Asia.

The following year$_{\text{[ID=t34, val=BC0491]}}$, having given clear warning of his intentions, Darius sent ambassadors to all the cities of Greece, demanding their submission.

To evaluate a temporal focus tracking algorithm, we could either check whether it selects the same reference expression that is specified in the gold standard as the correct choice, or we could check whether the reference time (i.e. the value) provided by the reference expression is the same as the value stored in the gold standard. In the case of the above example of the expression *the following year*, in the first option we would check whether a temporal focus tracking algorithm selected the expression with ID `t32`; in the second case we would expect the algorithm to find the value `BC0492`.

Of course both approaches require proper preparation of gold-standard data. In Table 6.10 we list what is provided in the existing gold-standard annotations. TIMEX2 provides neither the ID of the reference expression nor the value of the reference time. Although TimeML provides for each context-dependent expression the ID of the corresponding reference expression, this is not always enough for the purposes of evaluation: there are cases where several expressions can equally well function as the temporal focus for a given context-dependent expression. Consider the following example:

(6.35) The Confederate President was captured on *May 10*$_{\text{[ID=t12, val=1865-05-10]}}$ and the surrender of the Department of Florida and South Georgia happened *the same day*$_{\text{[ID=t13, val=[1865-05-10]}}$. Confederate Brigadier General "Jeff" Meriwether Thompson surrendered his brigade *the next day*$_{\text{[ID=t14, val=1865-05-11]}}$ and *the day following*$_{\text{[ID=t15, val=1865-05-12]}}$ saw the surrender of the Confederate forces of North Georgia.

Here, the expression *the next day* can be correctly interpreted when either of the two preceding expressions is chosen as the temporal focus. Given the existence of such cases, it is more straightforward to assess the value of the reference time, rather than compare the IDs, since the latter requires preparation of more complex gold-standard data with the multiple possible correct sources marked.

Since the corpora we use in our work are annotated using the TIMEX2 standard, which provides no support for evaluation of temporal focus tracking, we opt to use in our

Feature	TIMEX2	TimeML
Extent	yes	yes
Local semantics	no	no
Ref. time value	no	no
Ref. expr. id	no	yes
Global semantics	yes	yes

Table 6.10: An analysis of the information in the existing gold-standard annotations that may be useful for the evaluation of temporal focus tracking.

Doc.	# TIMEX2	# Req. Ref. Time
BC′	245	100 (40.8%)
BN′	134	60 (44.8%)
CTS′	213	37 (17.4%)
NW′	264	110 (41.7%)
UN′	320	44 (13.8%)
WL′	162	32 (19.8%)
Total	1338	383 (28.6%)

Table 6.11: The number of expressions in the sample of the ACE'05 Training corpus that require a reference time.

Doc.	# TIMEX2	# Req. Ref. Time
01	170	85 (50.0%)
02	265	89 (33.5%)
03	75	24 (32.0%)
04	147	36 (24.4%)
05	245	45 (18.3%)
06	149	34 (22.8%)
07	247	58 (23.4%)
08	175	35 (20.0%)
09	129	20 (15.5%)
10	57	0 (0.0%)
11	102	15 (14.7%)
12	98	27 (27.5%)
13	104	33 (31.7%)
14	71	39 (54.9%)
15	78	4 (5.1%)
16	63	43 (68.2%)
17	130	18 (13.8%)
18	110	11 (10.0%)
19	62	17 (27.4%)
20	106	63 (59.4%)
21	29	13 (44.8%)
22	69	30 (43.4%)
Total	2681	739 (27.6%)

Table 6.12: The number of temporal expressions in WikiWars that require a reference time.

experiments the values of reference expressions, rather than their IDs. To eliminate the possibility of errors introduced by incorrect extent recognition, we use gold-standard extents of temporal expressions as input to these experiments and we do not perform detection and recognition of temporal expressions; this concerns both the context-dependent expressions subject to the interpretation and those that provide reference times and may be context-independent. To eliminate the problem of error propagation (which occurs when for any reason a context-dependent expression is incorrectly interpreted and then functions as a reference time, thus providing an incorrect value for the interpretation of another context-dependent expression) we will also use the gold-standard values of the `VAL` attribute of those expressions that are selected by the temporal focus tracking algorithm as reference expressions. The results are reported as accuracies, i.e. the ratio of the number of cases when a correct reference value is determined to the number of all context-dependent expressions.

6.3.3.2 Evaluation Data

We run the experiments on the whole WikiWars corpus and six subsets of the ACE 2005 Training corpus; these subsets are prepared by randomly taking 20 documents from each of the six parts of the corpus containing documents from a different domain.[22] This ensures that we can look at the problem of finding the reference time in more detail with regard to the document type and source; for example, we already know that WikiWars does not contain any deictic expressions, [23] hence using the time-stamp of the document should yield no correct result.

As discussed previously, we need to extend the existing TIMEX2 annotations to provide each gold-standard annotation of a context-dependent expression with the key value of the reference time. We add a new attribute `refValue` containing the reference time at a granularity coarse enough, and not finer than necessary, to interpret the expression in question. For example, the gold-standard annotation of the expression *the next day* in Example (6.35) is:

(6.36) `<TIMEX2 val="1865-05-11" refValue="1865-05-10">`the next day`</TIMEX2>`

If the value of the reference expression chosen by the annotator for the above example was of a granularity finer than day granularity, we would omit all information finer than day granularity; for example, if the value was `1865-05-10T18:15` we would need to discard the unnecessary information expressed by `T18:15`.

We provide the `refValue` attribute only for point expressions whose interpretation is dependent on other temporal expressions. In particular, in our experiments we will not attempt to find the reference time for the following types of expressions: event-based expressions (e.g. *five days after the attack*), set expressions (e.g. *each day*), and references to the general past, presence and future (e.g. *weeks earlier*).

We will also omit underspecified parts of range expressions which as a whole are explicit, such as *31 May* in *31 May–1 June 1916*, or *17* in *1915–17*. We use the same approach for conjoined expressions involving ellipsis, e.g. *August* in *August and September 1914*. Similarly, expressions that really refer to one temporal entity, but which in TIMEX2 are annotated as two expressions, are not within our focus; this concerns, for example, *11 a.m.* in *11 a.m. on 11 November 1918*. Temporal expressions which appear in appositives to other temporal expressions are also not annotated with the `refValue` attribute; consider the following example:

(6.37) on *September 17, 1862*, *the bloodiest single day in United States military history.*

The second expression here (i.e. *the bloodiest single day in United States military history*) denotes the same temporal value as the preceding date; we exclude the second expression from the experiment on the grounds that apposition is a syntactic relation which can be used to determine that the two expressions refer to the same temporal entity, and it is not necessary to carry out temporal focus tracking for such cases. Other cases which are not annotated with `refValue` for our experiments include temporal expressions appearing as modifiers within noun phrases that also contain other fully-specified temporal expressions; for example, *Spring* in *Spring Offensive of 1918* and *February* in *February Peace Conference of 1861*.

We also removed from the datasets six expressions which were not context-dependent: these contained within their extents other context-dependent temporal expressions, where the matching of annotations in the evaluations was corrupted. The removal

[22]We will refer to these samples by adding a prime to the name of the dataset (e.g. CTS′ or ACE′) to indicate that this is a sample subset, not the complete corpus.

[23]Except for just a few cases which are erroneously placed in the WikiWars documents, for example in 07_IraqWar, and whose interpretation is problematic even for humans, since they are artefacts of bad writing.

of these expressions has no negative impact on the experiments with temporal focus tracking.

In Tables 6.11 and 6.12 we provide information on the number and proportion of context-dependent expressions present in the ACE samples and WikiWars documents, respectively. Interestingly, the ratios are very similar for the two sets: 28.6% for ACE′ and 27.6% for WikiWars. However, we note that for individual documents these ratios are very different, ranging in WikiWars from 0% (10_PunicWars) to 68.2% (16_SpanishCivilWar).

6.3.3.3 The Processing and Evaluation Procedure

To carry out the experiment we organize the processing in the following way. We read in the documents with gold-standard inline annotations, and we iterate over the annotations of temporal expressions from the beginning to the end of the document. For each context-dependent expression (i.e. where the `refValue` attribute in the gold standard is non-empty) we select a reference expression using a temporal focus tracking algorithm. The temporal value (i.e. the value of the `VAL` attribute) of the selected reference expression is read from its gold-standard annotation. The value is then adjusted, if necessary, to match the granularity expected in the `refValue` attribute in the gold-standard annotation of the expression being interpreted. The adjusted value is then stored in the `refValue` attribute of the output annotation; the extent of the output annotation is the same as the extent of the input annotation of the context-dependent expression being interpreted.

To compute the evaluation scores, we use the Corpus Quality Assurance tool provided with GATE; this reports the number of correct, incorrect (i.e. partially correct), missing and spurious matches. We configure the tool to evaluate the output based on the `refValue` attribute;[24] we calculate the accuracy by dividing the number of correct matches by the total number of context-dependent expressions found in the dataset.

6.3.4 The Experiments

6.3.4.1 The Algorithms

Based on the analysis of the problem and the approaches described in the literature, we identified a number of strategies for selecting the temporal expression to serve as the reference time for the interpretation of a context-dependent temporal expression. The solutions investigated here use time-stamps of documents and some variants of the recency-based model:

DCD For each context-dependent expression we use the time-stamp of the utterance (e.g. the document creation date) as the reference time.

MRP For each context-dependent expression we use as the reference time the most recent point temporal expression occurring before the expression in question that is not non-specific or too coarse in its granularity.

[24]Actually, the tool takes into account both the chosen attribute and extents, but since we use gold-standard extents to produce output annotations, all extents are correct and remain ‘transparent’ for calculating the performance measures, and we evaluate only the values in the `refValue` attribute.

Chronos This is an attempt to reimplement the approach used in the Chronos tagger developed by Negri and Marseglia (2005). It chooses between the DCD and MRP approaches depending whether the expression is considered deictic or not, respectively. The expressions considered deictic are: weekday and month names, *yesterday*, *today*, *tonight*, and those that contain any of the following key words: *this*, *last*, *next*, *past*, and *ago*.

DCD/MRP Before knowing the performance of the previous approach, we already suspect that the classification of deictic expressions used in Chronos is not always adequate. For example, weekday and month names can be used anaphorically as well as deictically. In practice, for some documents the question of whether an expression is deictic or anaphoric may be not important, and both the DCD and MRP methods may provide the correct reference time. However, since we already know that the documents in the WikiWars corpus contain a large number of month names and practically no deictic expressions, we expect that the Chronos approach will not perform too well on this corpus. Therefore, we use an improved classification of deictic expressions, consisting of *now*, *right now*, and any expressions containing the words *today*, *yesterday*, *tomorrow*, *tonight*; or ending with *ago*; or starting with *this*, *last*, *recent*, or *next*.

Sent+DCD/MRP Our intuition is that, before looking further back in the text, it may be worth first investigating the remainder of the sentence containing the temporal expression being interpreted. Consider the following example found in our data:

(6.38) On Germany's eastern front, the Axis defeated Soviet offensives in the Kerch Peninsula and at Kharkov and then launched their main *summer*$_{[1942]}$ offensive against southern Russia in *June 1942*, to seize the oil fields of the Caucasus.

Here, the correct reference time for *summer* is *June 1942*. To handle such cases, we augment the DCD/MRP heuristic to search for the reference time in the current sentence. First, we check whether the expression is deictic, in which case we use the document time-stamp. If the expression is not deictic, then we look for an appropriate reference expression towards the beginning of the sentence; if none is found, we then look in the part of the sentence from the expression being interpreted towards the end of the sentence, and if none is found here, we then use the most recent expression occurring before the beginning of the sentence. There is a clear similarity here with nominal anaphora, where a pronominal anaphor may have its antecedent later in the same sentence.

6.3.4.2 The Results

The results of applying the above methods are presented in Table 6.13. We evaluate each method on the individual datasets, i.e. the WikiWars corpus and the six subsets of the ACE corpus, but also on the combined set of the ACE samples (see the ACE′ column), and on all the documents considered together (the WW+ACE′ column). In each column we have marked with a grey background the best result obtained.

As anticipated, the DCD method did not work for the WikiWars corpus, achieving only 0.8% accuracy; all the expressions that obtained the correct reference time with this method were from the same document, 07_IraqWar, which described events from the year of the document time-stamp.[25] For the ACE documents considered together, the method provided the reference time with a reasonable accuracy of 91.6%, but we observe great variation across the different domains: from 75% for UseNet discussions to up to 98.3% for broadcast news.

Using the most recent point (MRP) temporal expression provides the correct reference time for 93.8% of cases in the WikiWars data and for 77.0% of cases in the ACE documents. Despite the drop in performance for the latter dataset, overall this method is more reliable than using DCD: the results for all the documents combined rose from 31.8% to 88.1%.[26] There are also fewer differences in performance between the various ACE domains than was the case with the DCD method.

For the ACE documents we get high accuracy with both the DCD and MRP methods (91.6% and 77.0%, respectively) because very often the entities denoted by most of the expressions found in the documents are located on a timeline close to the time-stamp of the utterance, and in consequence quite often no matter which expression we use as the reference expression, we get the correct reference time.

The approach used in the Chronos tagger, developed for participation in the ACE 2004 evaluations, provided the best performance of all the evaluated methods for the ACE′ dataset (93.7%). However, it provided the correct reference time in only 25.4% of cases for WikiWars. This is not surprising, given that all expressions containing month names were considered deictic and interpreted with respect to the document time-stamp.

When using our rules for the classification of deictic expressions, the DCD/MRP method provides the best compromise between the performance for the two distinct datasets; across all the documents used in our experiment, we achieve an accuracy of 90.1%. The lowest accuracy with this method, 74.5%, was obtained for newswire documents (NW′); however, 11 out of 28 incorrectly resolved reference times in this dataset concerned expressions from a single document.

Augmenting the DCD/MRP method with the search for reference time within the current sentence first turns out to do more harm than good, resulting in a slight deterioration in performance for both the WikiWars and ACE documents.

Finally, we also checked what would happen if we had some means of determining in advance what type of documents a dataset contains, so that the most appropriate method could be used. When for each of the six ACE datasets and WikiWars the reference time was determined with the method that performed best for it, the overall accuracy would be 94.4%. However, we note that implementing such an automatic detection mechanism may not be straightforward, since underspecified expressions may have the same lexical form whether used deictically or anaphorically.

6.3.4.3 Error Analysis

The low performance of the DCD method for the WikiWars corpus is related to the absence of deictic offsets (e.g. *tomorrow*) and underspecified expressions used deictically. While the method performs very well for the ACE′ dataset overall, we observe that for some of the domains the performance is much lower: for example, for UseNet discussions (UN) it drops to 75%. One of the reasons is that this dataset contains

[25]In Section 3.3.4 we discussed the issue of using deictic temporal expressions in this document.

[26]We note that since there are about twice as many context-dependent expressions in WikiWars than in the ACE′ dataset, the combined dataset WW+ACE′ favours the method that performs well for WikiWars.

Method	Dataset								
	WW+ACE′	WW	ACE′	BC′	BN′	CTS′	NW′	UN′	WL′
DCD	31.8	0.8	91.6	95.0	98.3	94.6	93.6	75.0	81.3
MRP	88.1	93.8	77.0	79.0	83.3	81.1	71.8	77.3	71.9
Chronos	48.8	25.4	93.7	93.0	95.0	86.5	96.4	90.9	96.9
DCD/MRP	90.1	92.7	85.1	92.0	90.0	83.8	74.5	86.4	90.6
Sent+DCD/MRP	88.1	92.4	79.6	84.0	88.3	70.3	70.0	84.1	87.5

Table 6.13: The accuracy of various temporal focus tracking methods; WW=WikiWars.

only a small number of context-dependent expressions (only 44), so every incorrectly-determined reference time results in a drop in accuracy of as much as 2.27%; the same issue concerns the weblog domain dataset (WL), which has even fewer context-dependent expressions. Given that the documents in these two domains bear some features of the narrative genre, the incorrect resolution of reference time in just a few cases is not surprising, but already has a huge impact on the reported accuracy for a given dataset.

Using the most recent point (MRP) expression turned out to have good accuracy for WikiWars; only 46 reference times were incorrectly determined, constituting an error rate of 6.2%. An analysis of the errors shows that in 35 cases the correct value was conveyed by the second most recent point expressions, and the third and fourth most recent expression would provide a correct value in yet another one case each. Most often the errors here occurred because the most recent expression was related to some additional information mentioned to the discourse, typically providing some background knowledge relating to the described events or concerning some events that occurred at some point in the future. Consider the following example:

(6.39) In *May 1944*, British forces mounted a counter-offensive that drove Japanese troops back to Burma, and Chinese forces that had invaded Northern Burma in *late 1943* besieged Japanese troops in Myitkyina. The second Japanese invasion attempted to destroy China's main fighting forces, secure railways between Japanese-held territory and capture Allied airfields. By *June*$_{[1944]}$, the Japanese had conquered the province of Henan and begun a renewed attack against Changsha in the Hunan province.

Here, we need to use the second most recent expression as the reference expression for *June*: between the descriptions of the events in May and June 1944 the text contains what we might think of as an embedded discourse segment which presents some earlier events from late 1943.

Other reasons for the immediately preceding point expression not being the reference time were as follows. In three cases the correct value is provided by the immediately following temporal expression, rather than the immediately previous; two instances concerned situations like that in Example (6.38); and one case was an underspecified reference to a year season appearing in a section heading, with the following paragraph providing the year. In two other cases only the document time-stamp provided the correct value; this was a result of the incorrect use of deictic expressions. Two other errors came from a multiply-conjoined expression, *ten, fifteen, and twenty years earlier*, which in TIMEX2 contains three annotations; all three expressions should be interpreted using the same reference expression, but in our set-up they were processed independently.[27] One expression, *Christmas*, concerned a decision to be made while preparing the gold standard for the experiment. A number of expressions in the document alternately referred to Christmas and January; we could either choose the January date to be the reference time and require the interpretation algorithm to detect the year border between the two values, or expect the temporal focus algorithm to find a coreferential link to an earlier mention of Christmas; since we chose the latter option, the selection of the most recent expression was not scored as correct. Finally, one expression (*the "good old days"*) was considered in the gold standard as co-referring to the 19th most recent expression; this case was, however, very difficult to resolve even for a human annotator.

6.3.5 Summary

In this section we have explored the problem of temporal focus tracking, i.e. obtaining the reference time for the interpretation of a context-dependent temporal expression.

We found that for some domains using the time-stamp of the utterance yields very good results; for example, for telephone conversations, broadcast news and newswire documents this solution provided the correct reference value in over 90% of cases; for broadcast news the accuracy was as high as 98.3%. The high results are directly related to the high proportion of deictic offset expressions and underspecified expressions that can be interpreted deictically.

For narrative documents, such as those found in the WikiWars corpus, where deictic expressions are practically absent, using the document time-stamp provided the correct reference time in fewer than one per cent of cases. Instead, the most previous point expression served as the correct reference time for 93.8% of context-dependent expressions. For the ACE sample dataset, this approach performed with 77.0% accuracy. The error analysis carried out for the WikiWars corpus showed that in 35 out of 46 error cases (76%) the correct reference time would be provided by the second most recent temporal expression. The development of a reliable heuristic for detecting these cases would improve performance by 4.7%. Such a heuristic could perhaps analyse whether the most recent expression denotes a time that is some distance from the times denoted by the second most recent temporal expression and the expression being interpreted.

If we had to use a single method for both types of documents, then automatically recognizing deictic expressions and using the document time-stamp for these cases, and using the most recent expressions for all the other cases, provides the best results for the two datasets combined, with 90.1% accuracy overall. Our set of rules for classifying deictic and anaphoric expressions provided a marked improvement over the heuristics used in the Chronos tagger, which in our reimplementation scored only 48.8% on the two datasets combined.

If an oracle was available to specify which strategy to use for a particular dataset,[28] the accuracy for the data we used would improve to 94.4%.

[27]This could have been addressed in the experimental design by removing from the gold standard the annotations of *ten* and *fifteen*.

[28]We note that in practice it could be the user who performs the role of the oracle by specifying in an input parameter which strategy should be used; this is how it is done, for example, in the case of the recently-developed HeidelTime tagger (see the demo at `http://dbs-projects.ifi.uni-heidelberg.de/heideltime`).

6.4 The Interpretation of Bare Weekday Names

In this section we explore the development of an algorithm for the interpretation of bare weekday names, such as *Saturday*, in particular with regard to determining the direction of offset to be used in the temporal interpretation of these expressions: in essence, how can we determine whether the day referred to is in the past or in the future with respect to the reference time?

6.4.1 What's the Problem?

Many temporal expressions in text are not fully specified (explicit), and require contextually-sourced information in order to determine their correct interpretation. In some cases, it is sufficient to determine the reference time. Consider the following examples:

(6.40) a. *three days ago*
b. *last Monday*
c. *two weeks later*

Once we know the reference time, calculation of the temporal location referred to in each of these cases is straightforward, since the temporal expressions themselves explicitly indicate what we will call **the direction of offset** (here, respectively, past, past and future). However, in other cases there is no explicit indication of the direction of offset from the reference time. This is most obviously the case when bare weekday names are used,[29] as in the following example:

(6.41) Jones met with Defense Minister Paulo Portas on *Tuesday* and will meet Foreign Minister Antonio Martins da Cruz before leaving Portugal *Wednesday.*

Here, the proper interpretation of the references to *Tuesday* and *Wednesday* requires at the least a correct syntactic analysis of the sentence, in order to locate the **controlling verb** for each weekday name. The tense of this verb can then be used to determine the direction—either in the past or in the future—in which we need to look to establish the fully-specified date referred to. In the case of Example (6.41), this means determining that *Tuesday* is in the scope of the verb *met*, and that *Wednesday* is in the scope of the verb group *will meet.* But there are three problems with this. First, especially when the sentences considered are complex, there is a non-negligible likelihood that the analysis returned by a parser may not be correct, as we noted in our discussion of extent recognition in Chapter 5. This is especially the case when the sentences in question contain structures such as prepositional phrases: the attachment of these is notoriously a source of ambiguity, and they just happen to often be the hosts to temporal expressions. Second, even if a parser provides the correct analysis, parsing technology is still computationally expensive to use when processing very large bodies of text; if we are interested in time-stamping events described in significant volumes of data, we would prefer to have a faster, more heuristic-based approach. Third, it turns out that there are cases where even the controlling verb does not provide sufficient information to determine the direction of offset, as demonstrated by the following pair:

[29]Generally, this problem may occur in case of any underspecified expression, be it a bare month name (*March*), a bare season name (*winter*), a date (*the 17th*), or even a decade (*'30s*). However, our experience and the error analyses found in the literature suggest that the problem of determining the direction of interpretation concerns particularly bare weekday names, hence the limitation in the experiments presented here.

(6.42) a. We can show you some pictures on *Monday.*

b. We can show you some pictures from *Monday.*

Despite the same verb group used in the two sentences, *can show*, in Example (6.42a) the direction of interpretation is future, and in Example (6.42b) the direction of interpretation is past.

We are interested, therefore, in determining whether some heuristic method might provide good results.

6.4.2 Related Work

The literature contains a number of descriptions of work concerning interpretation of weekday names accompanied with some 'signal words', such as *last* or *following*; this includes the work of Filatova and Hovy (2001), Negri and Marseglia (2005), Ahn et al. (2005b) and Han et al. (2006b). However, we are interested in the interpretation of bare weekday names, in which case such clues are not available.

Most closely relevant to the work described in this section are the approaches described by Baldwin (2002), Jang et al. (2004) and Mani and Wilson (2000b). Since we have re-implemented versions of these algorithms for the work described here, we leave description of these to Section 6.4.4. These approaches all use rule-based heuristics.

A different approach was taken by Ahn et al. (2007), who describe a system using a classifier based on support vector machines. This algorithm was used to determine the direction of all context-dependent temporal expressions, not just the names of weekdays. They used three sets of features:

1. Character type patterns, lexical features such as weekday name and numeric year, a context window of two words to the left, and several parse-based features: the phrase type, the phrase head and initial word (and POS tag), and the dependency parent (and corresponding relation) of the head.
2. The tense of the closest verb (w.r.t. dependency path), the POS tag of the verb, and the POS tags of any verbal elements directly related to this verb.
3. Features comparing year, month name and day name of a temporal expression to those of the document creation date.

Their experiments demonstrated that the third set was the most useful.

6.4.3 The Experimental Corpus and Set-up

For this work we used the ACE 2005 Training corpus. Table 6.14 shows the distribution of bare weekday names (as TIMEX2 counts) in the corpus across the various genres represented.[30]

For the work described here, we used only those documents in the corpus that contained at least one weekday name; all subsequent analysis makes use only of the gold-standard annotations of the bare weekday names in these documents, thus significantly reducing corpus processing time. This results in a total of 367 instances, once errors (of which there are quite a few) in the gold-standard annotations have been repaired. We made the following changes to the gold-standard data:

[30]Recall that: BC = Broadcast Conversations; BN = Broadcast News; CTS = Conversational Telephone Speech; NW = Newswire; UN = UseNet Newsgroups; and WL = Weblogs.

Corpus Part	Num. of Docs	Num. of TIMEX2	Num. of Bare Weekday Names (BWNs)	Percent of all TIMEX2	Num. of Docs with BWNs	Num. of BWNs per one doc (for docs with such names)	Num. of BWNs per one doc (for all docs)
BC	60	626	7	1.91%	4	1.75	0.12
BN	226	1455	31	8.47%	25	1.24	0.14
CTS	39	409	2	0.54%	2	1.00	0.05
NW	106	1235	292	79.56%	102	2.86	2.75
UN	49	741	3	0.81%	3	1.00	0.06
WL	119	1003	32	8.72%	19	1.68	0.27
Total	599	5469	367	100%	155	2.37	0.61

Table 6.14: Bare weekday names in the ACE 2005 Training Corpus.

- One day name had been missed by the annotators; we added this.
- Some 40 values were corrected from the format YYYY-Wnn-m to YYYY-MM-DD: although both are in some sense correct, the TIMEX2 guidelines indicate that the second is the preferred form.
- Eight cases where the incorrect value had been provided by the annotators were corrected.

For simplicity, for all weekday names interpreted in the experiments we take the temporal focus to be the document creation date; as we showed in the previous section, this heuristic provides reasonable performance for the ACE data.

6.4.4 Evaluated Approaches

We implemented and evaluated a number of both simple and more complex approaches to determining what date is meant in a text when a bare weekday name is used. These methods, described below, can be divided into two main classes: (a) 7-day window based, and (b) tense-analysis based. Our new algorithm is a hybrid solution that incorporates ideas from both of these approaches.

6.4.4.1 Baselines

Our baselines are motivated by the observation that days referred to by bare weekday names are typically temporally close to the temporal focus.
Past 7-day Window (inclusive): This baseline looks for the specified day in a 7-day window whose last day is the temporal focus. In other words, day names are always assumed to refer to days in the last week, including the ‘day of speaking’.
Past 7-day Window (exclusive): This is the same as the approach just described, except that we look for the referred-to day in the week leading up to but not including the ‘day of speaking’.
Future 7-day Window (inclusive): This is the future-oriented version of the first approach described above: we look for the specified day in a 7-day window whose first day is the temporal focus. This assumes that all day name references are to the present

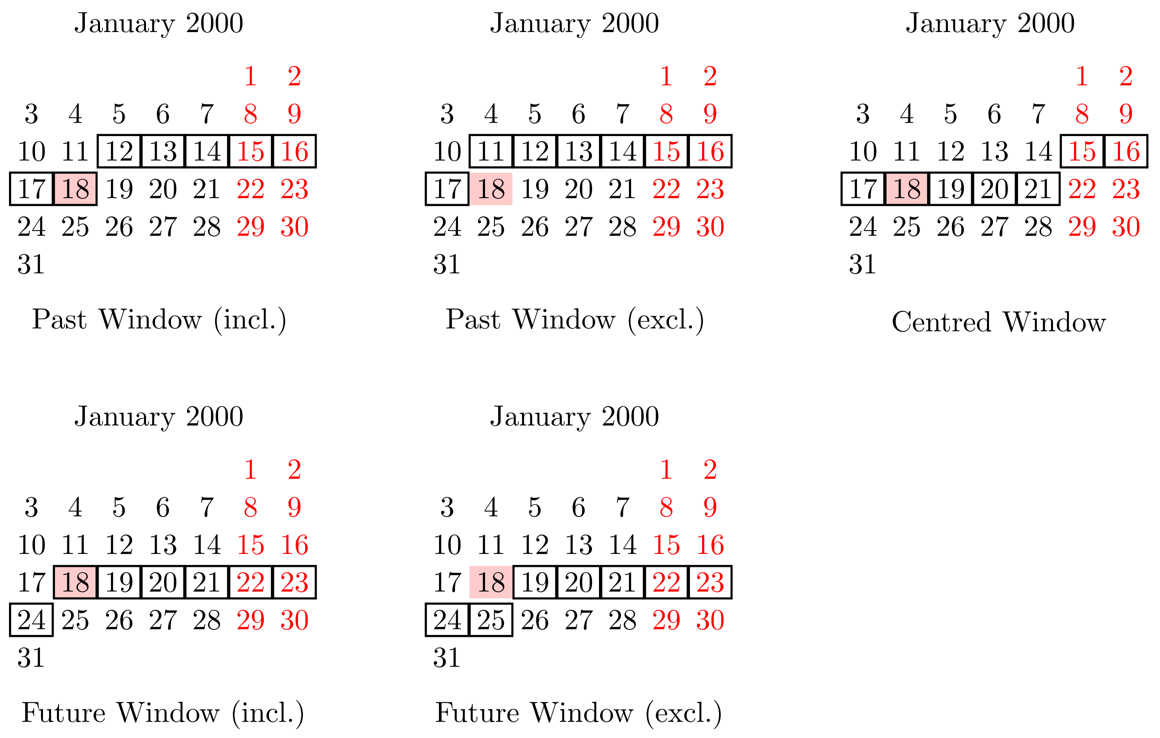


Figure 6.5: Window-based heuristics for the interpretation of weekday names.

or future.[31]

Future 7-day Window (exclusive): In this case the 7-day window starts on the day following the ‘day of speaking’.

The windows selected by the above heuristics are presented in Figure 6.5. The date marked with a shaded background (18th January) is the reference date; the dates being part of the window for the given reference date are marked with rectangle frames.

6.4.4.2 Algorithms

Baldwin’s 7-Day Window This algorithm was presented by Baldwin (2002) and later also used by Jang et al. (2004). It is similar to our window-based baselines, but in this case the temporal focus is the middle day of the 7-day window; see the calendar titled ‘Centred window’ in Figure 6.5. This approach was used by Baldwin after observing that 96.97% of weekday name expressions in her English corpus referred to dates within such a window. Suppose we have the following sentence in a document with the time-stamp of 2003-06-16 (a Monday):

(6.43) Police arrested her in Abilene, Texas, *Saturday* where she had moved with a friend June 2.

The 7-day window then spans from Friday (June 13) to Thursday (June 19). The reference to *Saturday* is assigned (correctly) the value of the second day in the window, i.e. 2003-06-14. Note that this method will deliver the wrong result when the referred-to day actually falls further than three days either side of the temporal focus. Suppose,

[31]An informal check of email data drove Han et al. (2005) to use the simple strategy of always assuming that weekday names refer to days in the future.

for example, we have the following sentence in a document written on 2005-01-01 (a Saturday):

(6.44) We got into Heathrow on *Monday* morning.

Here the 7-day window spans from Wednesday to Tuesday, and so the reference to *Monday* will be assigned the incorrect interpretation 2005-01-03 instead of 2003-12-27, which is outside the window.

Mani and Wilson's Tense Estimation In the system presented by Mani and Wilson (2000b), weekday name interpretation is implemented as part of a sequence of interpretation rules for temporal expression interpretation more generally. This algorithm attempts to establish the tense of what we have called the controlling verb in the following way. First, it looks backwards from the temporal expression in question to any previous temporal expression in the sentence, or if there is none, to the beginning of the sentence. If no verb is found here, then it looks between the temporal expression and the end of the sentence; and if a verb is still not found, then it looks in front of any preceding temporal expression found back to the beginning of the sentence. If the verb found is in past tense, the direction of offset is assumed to be backwards; if the tense is future, then the forward direction is used. If the verb found is in present tense, then the temporal expression is passed to a further set of interpretation rules, which check for things like the occurrence of lexical markers such as *since* or *until.*[32] For instance, in Example (6.41), repeated below, the algorithm would correctly pick *met* for *Tuesday* and *will meet* for *Wednesday*, interpreting Tuesday as a day in a past and Wednesday as a day in future.

(6.41) Jones met with Defense Minister Paulo Portas on *Tuesday* and will meet Foreign Minister Antonio Martins da Cruz before leaving Portugal *Wednesday.*

However, this approach will not correctly interpret Example (6.45):

(6.45) Still a decision has to made on what, if any, punishment he will face in the wake of that incident *Tuesday* night.

In this case, the wrong verb will be identified, and the direction of offset will be incorrect.

Simple Tense Estimation As an alternative to Mani and Wilson's approach, we also implemented a much simpler tense estimation heuristic. This checks whether the sentence contains any tokens with the VBD (i.e., past tense) part of speech tag;[33] if one is found, then the direction of offset is assumed to be backwards, and if not, then we use the forward direction. In the case of Example (6.41), this will assign the correct value to *Tuesday*, but the wrong value to *Wednesday.*

[32]We have reimplemented this algorithm based on the description given in the cited paper, but some details are unclear, so we acknowledge that the original implementation might produce slightly different results.

[33]Where POS tags are required in our algorithms, we used Mark Hepple's part of speech tagger, an implementation of which is available as a plugin for the GATE platform.

Tense	Example	Direction
Past Simple	I *wrote* a paper on *Monday.*	Past
Present Perfect	I *have been writing* a paper since *Monday.*	Past
Present Continuous	I *am flying* to New York on *Monday.*	Future
Future Simple	I *will write* a paper on *Monday.*	Future
Bare Past Participle	The draft *written* on *Monday* was useless.	Past
Modal Verb	I *should finish* the paper on *Monday.*	Future
Modal Verb + *have*	I *should have submitted* the paper on *Monday.*	Past

Table 6.15: Tense interpretation rules.

Dependency-based Tense Determination The two previous algorithms attempt to determine the controlling verb using very simple heuristics. Of course, a more reliable way of determining the controlling verb is to use a parser. We used the Stanford parser's dependency information output (see (de Marneffe et al., 2006)) to find the controlling verb of a weekday name in a sentence. This algorithm does this by traversing the resulting dependency tree from the node containing the weekday name to its root until a verb is found, and then following further dependencies to identify the whole verbal sequence.

A Hybrid Algorithm Heuristic methods for determining tense are risky, especially as the distance between the controlling verb and the temporal expression increases. We therefore propose a hybrid approach that attempts to leverage both tense estimation approaches like Mani and Wilson's, and Baldwin's window-based approach. This algorithm was developed on the basis of an error analysis of the results of using Baldwin's algorithm. It embodies a two-step approach, where we first look only in the very local environment for clues as to the tense of the controlling verb, then fall back on Baldwin's algorithm if no such evidence is found close by.

First, we check if the temporal preposition *since* appears immediately in front of a weekday name; if so, the direction of offset is assumed to be backwards; otherwise, the algorithm looks for any verbs in a window of three tokens before and three tokens after the temporal expression. If a verb is found, then its tense is used to determine the direction (using the same rules as in Mani and Wilson's approach). If no verb is found, then a 7-day window with the reference time as the middle day is used, just as in Baldwin's algorithm.

Voting This algorithm uses a voting mechanism over the output of Baldwin's, Mani and Wilson's, and the Dependency-based Tense Determination algorithms. If all values are different (no majority) then Baldwin's result is used.

Once the verb group is found by any particular algorithm based on tense determination, it needs to be analysed to determine what its tense is; this information is then used to determine the direction of offset. The tense interpretation rules are summarized in Table 6.15.

Algorithm	Errors	Correct	Correct (%)	Time [sec.]
Past 7-day Window (Inclusive)	51	316	86.10%	79.9
Past 7-day Window (Exclusive)	240	127	34.60%	79.7
Future 7-day Window (Inclus.)	129	238	64.85%	79.2
Future 7-day Window (Exclus.)	316	51	13.90%	79.4
Sentence Tense Estimation	38	329	89.65%	80.6
Dependency-Based	29	338	92.10%	616.5
Mani and Wilson's	27	340	92.64%	80.9
Baldwin's 7-day Window	21	346	94.28%	79.4
Voting	16	351	95.64%	636.1
Hybrid	15	352	95.91%	80.2

Table 6.16: The accuracy and processing times for the interpretation of bare weekday names.

6.4.5 Results

Table 6.16 presents the results achieved with each of the algorithms. The 51% difference between the inclusive and exclusive baselines is indicative of the fact that, in this data, in over 50% of cases the correct date was in fact the document creation date. This phenomenon is due to the large proportion of newswire data in the corpus; in this genre, it is common to use the weekday name even when reporting on events that happen on the same day as the reporting takes place. Also of note is that the best performing baseline, 'Past 7-day window (inclusive)', achieves 86.10% accuracy despite its being an extremely naive approach.

All the algorithms tested here performed better than the baselines. The best performing algorithm was the Hybrid method, which made 15 errors, resulting in an accuracy of 95.91%; the Voting method came second with 16 errors. Baldwin's 7-day window algorithm correctly interpreted 94.28% of weekday names. The big advantage of this algorithm, along with all the baselines, is their complete resource independence: they do not use any parsers or POS taggers.

Perhaps surprisingly, Mani and Wilson's tense estimation heuristic was more effective than tense determination based on a dependency parse tree; this reinforces our earlier point about the risks of using parsers. It is also important to note that there are huge differences in execution time for parser-based approaches. In our set-up the dependency based method executed in 616 seconds, while all other methods executed in around 80 seconds; the differences in time is huge despite the fact that we run the parser only for those sentences that contain bare weekday names.

6.4.6 Error Analysis

The Hybrid Algorithm achieved the best accuracy of 95.91%, which corresponds to 15 error cases. These were as follows:

- Eight cases where there was no verb found in the three-token neighbourhood of the temporal expression; in these cases the 7-day window method was used, but this did not find the correct value.

- Three cases where the algorithm identified a verb that was not the controlling verb; for example, it picked *will meet* instead of *met* to interpret *Tuesday* in the sentence given in Example (6.41).
- Two cases where the document creation date was very misleading (see below).
- Two cases where past tense was used to talk about plans for the future which were subsequently cancelled, as in *discussions were scheduled to end Friday, when Kelly was to fly. . . .*

In 204 cases the algorithm interpreted the weekday name based on a verb found in the three-token neighbourhood; and in 163 cases it used the fallback 7-day window strategy. Since the Hybrid Algorithm was built as an extension of Baldwin's method, it is worth knowing whether there were any cases where the original 7-day window method got the correct value and the Hybrid Algorithm got it wrong. There were six such cases:

- Two of them occurred for documents with a misleading document creation date. In a typical example, a document with the time-stamp 17-04-2004 (a Thursday) contained the sentence 'Malaysia's Appeal Court *Friday* refused to overturn the conviction . . . '. As the document time-stamp was used as the temporal focus, *Friday* was interpreted as a day in the past, when in fact it was the day after the time-stamp.
- The other two cases demonstrate a weakness in our approach, exemplified by the sentence given in Example (6.41): here the algorithm incorrectly uses the verb group *will meet* when interpreting *Tuesday.*
- The remaining two cases were cases where the verb groups *were scheduled to end* and *scheduled to begin* were used to talk about future events.

In these last cases, the controlling verb is an infinitive, and there is no way, in the absence of either world knowledge or a much more sophisticated analysis of the text, of determining whether the scheduled event is in the past or the future. Sentences like these are a particular problem for Mani and Wilson's algorithm, where a significant number of misinterpretations involve sentences in which the past tense is used to talk about subsequently-changed plans for future, as in the following:

(6.46) A summit between Sharon and his Palestinian counterpart, Mahmoud Abbas, had been planned for *Wednesday* but was postponed . . .

Here, this utterance could be legitimately produced both before and after the Wednesday in question, so no simple algorithm will be able to determine the direction of offset.

6.4.7 Summary

In this section we have investigated the problem of the interpretation of bare weekday names in texts, and presented a new heuristic which extends Baldwin's (2002) approach. Our evaluations on a widely-available dataset show that our Hybrid Algorithm was the best performing algorithm, achieving an accuracy of 95.91% with only 15 errors out of 367 instances. Baldwin's 7-day window heuristic performed with 94.28% accuracy, making 21 errors.

6.5 Other Challenges and Problems

In this section we discuss a number of issues which a temporal expression tagger must address if it is to claim truly broad coverage.

6.5.1 Calendar Arithmetics

Calculating the value of offset expressions (e.g. *two months later*, *yesterday*, *five hours ago*) requires carrying out calculations on dates and times using what we refer to as **calendar arithmetics**. Although we consider only two basic operations, addition and subtraction, it turns out that there are pitfalls related to characteristics of the underlying Gregorian calendar.

The basic issue is the proper handling of these operations across the boundaries of temporal units such as days, months and years. It may happen that an operation on hours or minutes crosses the boundary of a day; for example, *five hours ago* uttered at three in the morning denotes 10pm the previous day. Similarly, adding or subtracting days may cross the boundary of a month or a year. Calendar arithmetics must be implemented properly to detect all such situations and provide the correct results. Crossing the boundaries of temporal units must also account for leap years and different length of months; issues arising from the presence of a leap year may appear even when operating on processing offsets at the granularity of minutes or seconds (e.g. *25 minutes later* may cross the end of February).

Another issue appears with the interpretation of expressions like *a month later*. Although such an expression may be used as an approximation of a temporal distance between two events, we would typically interpret this as referring to the same date in the following month; for example, when uttered on the 18th January, then we can assume that the 18th February is meant. However, when uttered on the 31st January, we may instead choose to interpret the expression as referring to the last day of February.[34]

Further complexities are involved when dealing with daylight savings time (DST). To interpret, for example, the expression *18 hours later*, we must check whether the offset extends across a point in time where the change between the standard time and daylight savings time occurs. A difficulty here is that different countries adjust to daylight savings time on different dates, and they may also change these dates from one year to another. The tagger therefore requires access to up-to-date information about when DST starts and ends in particular years and countries. Moreover, to apply this information, it must also be aware of which geographic region the text assumes.

6.5.2 The Interpretation of Some Underspecified Expressions

Although in general the interpretation of underspecified expressions can be carried out just by filling the missing elements in the representation of local semantics with a value of proper granularity from the reference time (e.g. the year when interpreting a bare month name), things are more complex when the expected temporal value is from a different calendar unit than the value of the reference expression. Consider the following examples:

[34]TIMEX2 advises using the granularity of a month for such cases, so the interpretation would result in a value denoting February without referring to a day; but this is not an ideal solution for any application relying on precise temporal information extraction.

(6.47) a. We planted 50 trees in *autumn 1985* but many did not survive *the winter.*

b. The treaty was signed in *November*, but it was not respected until *February.*

c. I went to bed at *11pm*, but at *about 2:30am* I woke up and could not fall asleep again.

d. We are in *year 2009* but the music from *the '60s* is still most wanted by some people.

In all these cases there is a shift from one calendar unit to another; for example, in sentence (6.47d), the expression *'60s* refers to the 1960s rather than the 2060s. Effectively, similar calendar arithmetics must be carried out as for offset expressions, although in this case the offset is never more than one unit.

The problem concerns not only references to times close to unit borders, such as the months of December or January; for example, in one of the WikiWars documents we find *May* being mentioned in reference to *26 October 1955.* Here, the interpretation algorithm must somehow reason (e.g. based on the tense or some key words) that May 1956 is intended, rather than May 1955. We might view the problem here as being similar to the interpretation of bare weekday names as discussed above.

6.5.3 The Twelve-Hour Clock

The internationally-accepted convention for expressing time is the 24-hour clock, where each day starts at 00:00, eventually passes through 23:59 and finishes at 00:00 to close the daily cycle. However, in many countries, especially those which are English-speaking, an alternative popular convention is the 12-hour clock, in which a day is divided into two periods of 12 hours.[35] The period (a half of a day) which the given time concerns is marked with *a.m.* or *p.m.* designators, where the former stands for the Latin *Ante Meridian* (meaning *before midday*, i.e. before noon) and the latter stands for Post Meridian (in Latin meaning *after midday*, i.e. after noon). Given these meanings of *a.m.* and *p.m.*, both the expressions *12 a.m.* and *12 p.m.* should refer to midnight; however, many people, incorrectly,[36] use these designators differently: most often, *12 a.m.* refers to midnight, and *12 p.m.* refers to noon. Given the popularity of such a use of these designators, a temporal expression tagger should be able to correctly recognize the intended meaning.

Often the half-day designators are not used, thus making the expressions under-specified. Consider the following examples:

(6.48) a. I will go shopping at *five pm.* I should be back at *six o'clock*$_{[pm]}$.

b. I go to uni at *seven am.* I always have a number of different tasks to do throughout *a whole day*, and I catch the train back home at *nine*$_{[pm]}$.

c. I usually come back home at *seven o'clock*$_{[pm]}$.

d. We will depart on *17 Jan 2009, 5:15*$_{[am]}$. We will stop for *morning* tea at *10:20*$_{[am]}$. At *11:00*$_{[am]}$ you will receive booklets about the region. We will

[35]There is also the 6-hour clock, used traditionally in Thailand, in which a day is divided into four periods of six hours.

[36]What should be really used in text, to be unambiguous, is *12 noon* and *12 midnight.* Some companies, e.g. railroads and airlines in USA, also tend to use *12:01am* to mark the beginning of a day, and *11:59 pm* for the end. This is done particularly for the purpose of setting up a timetable or signing legal contracts.

have another break at *1:45*$_{[pm]}$ for lunch. We should get to the caves at *two*$_{[pm]}$. We will start driving again at *nine*$_{[pm]}$.

In Example (6.48a) to interpret *six o'clock* we can use the same day half as in the previous expression *five pm.* This approach would not work for Example (6.48b), where it must be reasoned that *nine* refers to 9pm since the sentence mentions that there are some activities throughout the day. In Example (6.48c) there are no temporal expressions in the context that could suggest that *nine o'clock* refers to a time in the evening; this could be evidenced only by the type of the event, coming back home, but even this is not a safe bet: somebody may be working night shifts and come back home at seven in the morning. Finally, when interpreting the expressions in Example (6.48d) we would need to analyse the contextual expressions (*morning*) and the kinds of events mentioned (*lunch*), and to track the time through the narrative, assuming that it presents the story chronologically.

If a time zone is used with the time, e.g. *11:00 AEST*, this may suggest that a formal format is being used, where no ambiguity is allowed; this may be taken to imply the use of the 24-hour clock, in which case this expression refers to eleven in the morning.

We note that to resolve such ambiguities, Han et al. (2006b) used the strategy of choosing the point which is closer to the temporal focus; for example, if the temporal focus was 1pm, the interpretation of *three o'clock* would be 3pm, not 3am. However, we observe that this approach would not work for the expression *nine* in Example (6.48b).

6.5.4 Ambiguous Triggers

In Chapter 5 we defined the trigger of a temporal expression to be a lexical item whose presence is a strong indicator that there may be an instance of a temporal expression in the text. Some triggers, for example weekday names, are very likely to signal an occurrence of a temporal expression. But other triggers are more ambiguous; for example, *spring* and *fall* may refer to a year season, or be used in one of their other, non-temporal, meanings. Compare the following two examples:

(6.49) a. We travelled around Europe in *the fall of the following year.*

b. *In the end we will see *the fall of the mighty empire.*

In the first sentence there is a temporal expression, but in the second one, there is none. Note that syntactic information cannot be used here to disambiguate the use of the trigger. Therefore, some sort of word-sense disambiguation is required to avoid false positive (spurious) or false negative (missing) matches; for example, a semantic tagger annotating tokens with WordNet synsets would provide sufficient criteria to decide whether the trigger denotes a temporal expression or not.

A specific kind of ambiguity concerns lexical items which have several time-related meanings. For example, *today* may mean 'the day that includes the present moment' or 'the present time or age'. Since these two meanings refer to different times, for two different uses of *today* in text there may need to be provided different values in the annotations. A similar issue occurs with the use of *now* or *time*, as shown in the following examples:

(6.50) a. We need to leave *now.*

b. **Now*, don't blame me.

(6.51) a. I will have grandchildren by *the time they finish building the underground in my city.*

b. * *This time* there is no space for a mistake.

Here the differences are even greater than in the case of *today*, since in Examples (6.50b) and (6.51b) there might be considered to be no temporal expressions at all (this is the position taken by the TIMEX2 and TimeML guidelines).

One approach that could be taken to decide which sense of an ambiguous trigger is being used in a text is to train a machine-learning classifier. For example, Mani and Wilson (2000b) experimented with disambiguating between the generic and specific use of *today* and, depending on the algorithm and features used, obtained up to 79.8% accuracy, against a majority-class baseline of 66.5%.

6.5.5 Providing an Anchor for a Duration

When interpreting an unanchored duration expression, such as *two weeks*, a temporal expression tagger producing TIMEX2 annotations needs to find in the remainder of the document a temporal anchor (i.e. the starting or the ending point) for the period. Therefore, providing the anchor comes down to finding a temporal expression whose value should become the value of the anchor. This is conceptually similar to temporal focus tracking carried out to interpret context-dependent expressions, with the difference that the value of selected expression is not used to calculate the global value of a point expression, but to provide the anchor of the period. Of course the actual algorithm for finding the anchor would need to be quite different from an algorithm for temporal focus tracking.

One of the challenges is to distinguish which durations should be anchored, and which should not. Example (6.52) presents a sentence from an article about some legal conflicts between participants of the America's Cup sailing challenge concerning the purchase of yacht design secrets.[37]

(6.52) He says the syndicate also received details of the revolutionary Millennium Rig, which took TNZ *three years* to develop.

The sentence mentions that building the yacht, whose design secrets were illegally sold, took three years; but the article does not provide any details about when the construction started or ended, as these are not important facts for the story. In consequence the period must remain unanchored. However, if the same sentence was used in another document, it would not be unreasonable to expect the text to provide an anchor for this period.

Consider also the text in Example (6.53) which is sourced from the Wikipedia article about the Premier League in England.[38]

(6.53) In response to concerns that clubs were increasingly passing over young British players in favour of signing less-expensive foreign players, in *1999*, the Home Office tightened its rules for granting work permits to players from countries outside of the European Union. Currently a non-EU player applying for the permit must have played for his country in at least 75% of its competitive 'A' team matches for which he was available for selection during *the previous two years*, and his country

[37]See `http://archives.cnn.com/2002/WORLD/sailing/02/12/nz.spt/index.html`.

[38]See `http://en.wikipedia.org/wiki/Premier_League`.

(a)	A man spent *15 years*$_{[start,1993]}$ piecing together 2,000 fragments of love letters to his late wife which she tore up when she caught someone reading them.
(b)	Ted Howard, 82, wrote 98 letters to Molly during *the seven years*$_{[start,late1940s]}$ he spent travelling Europe as a farm worker. When she found someone reading them in *1953* she tore them up. Mr Howard, of Ramsey in Cambridgeshire, began putting the pieces back together in *1993* and has just completed the notes, *three years after his wife died.* He wrote the love letters on hotel writing paper as he travelled the UK, Ireland, France and Holland in *the late 1940s* and *early 1950s.*

Figure 6.6: An example of a text where finding anchors of periods is a non-trivial task.

must have averaged at least 70th place in the official FIFA world rankings over *the previous two years.*

In this text we have two occurrences of a period expression, *the previous two years.* A cursory reading of the text may suggest that these expressions refer to a two year period finishing in 1999, the year mentioned earlier in the text. These expressions, however, are anchored on the hypothetical event of applying for the permit (i.e. they concern certain rules and are used generically); therefore, no specific temporal value can be provided as the anchor.

Finding the anchor is often not a trivial task. In some cases, as shown in Example (6.54), the anchor may be in the same sentence as the period expression.

(6.54) The show starts at *7pm* and is *90 minutes* long.

But quite often a sophisticated analysis of the discourse must be carried out. Consider the text in Figure 6.6, which is a fragment of an article from BBC News;[39] part (a) is the headline and part (b) contains the first four sentences of the article. In the headline we have the period expression *15 years.* Its starting anchor is 1993, which is to be found in the third sentence of the body of the article. The challenge in finding this anchor is that, before the expression *1993* appears in the text, there is another year expression referring to 1953. The correct identification of the anchor is only possible by matching the semantics of the two sentences which talk about piecing together fragments of letters: one sentence says how long it took, the other one says when it started. Doing this might be attempted with some simple heuristics based on, for example, word overlap (here, *piece* is once used as a verb once as a noun) or occurrence of phrases with similar meanings (e.g., *piecing together 2000 fragments* and *putting the pieces back together*). Note also that in this text there is another period expression, *the seven years*, which is also not to be anchored with 1953, but with the late 1940s, appearing later in the text. Again, comparing *travelling Europe* with *travelled the UK, Ireland, France and Holland* provides a hint that the two sentences are talking about the same period.

6.5.6 The Interpretation of Event-Based Expressions

Determining the actual point in time referred to by an event-based temporal expression, such as *the third day of the battle*, is difficult, since it requires an analysis of the text to

[39]See `http://news.bbc.co.uk/2/hi/uk_news/england/cambridgeshire/7506169.stm`.

first identify the underlying event, and to determine the time (a point or period) of that event. In other words, both event extraction and event time-stamping are involved.

While in many cases the time of the event is mentioned in the document, sometimes the document talks about events which the target reader is assumed to know about; for example, in a genre such as news, these could be important events in the history of the world (e.g. the start of WWII or the fall of the Berlin Wall) or some facts from the lives of famous people (e.g. the date of the death of Michael Jackson). Depending on the domain, the assumed-to-be-known events may be of course very different.

The problem that arises is how to provide the temporal value of an event-based expression if the document does not provide a time-stamp for the underlying event. One solution is to use some form of gazetteer of well-known world events. But maintaining such a resource would take significant ongoing effort, so a more dynamic approach might be preferred.

Assuming that we are talking about a generally-known event, one idea would be to use a web search engine to find the time associated with the event. For example, to interpret the expression *two days after Michael Jackson died* we need to know the time-stamp for the event denoted by *Michael Jackson died.* The top four results from Google when queried for the phrase 'Michael Jackson died' provide us with the following snippets of web pages:

(6.55) a. *25 Jun 2009* – We've just learned Michael Jackson has died. He was 50.

b. *26 Jun 2009* – Michael Jackson 2009: The year Michael Jackson died...

c. On *June 25, 2009*, American singer Michael Jackson died of acute propofol intoxication ...

d. *25 Jun 2009* – Entertainer Michael Jackson died after being taken to a hospital on Thursday having suffered cardiac arrest, according to the Los Angeles ...

In two cases the date returned as part of the results is the date of publication of the news, but this coincides with the date of the event—a common feature in the domain of news, of course. Although one of the snippets provides a different date than the others, we can find the correct date here by either using a voting mechanism or choosing the top-ranked result. Of course, it remains to be seen whether this approach would provide accurate results more generally.

We found there are cases in the WikiWars corpus where the method could improve the performance of the interpretation process. For example, in the 04_AmRevWar document we find an expression that cannot be reliably interpreted using only the text of the document: *Three weeks after the siege of Boston began.* It is not clear from the text which of the previously mentioned dates should be considered the beginning of the siege. If we searched on the web for 'the siege of Boston began', the top-ranked result provides the following sentence:

(6.56) The siege of Boston began on *April 19, 1775*, when, in the aftermath of the Battles of Lexington and Concord, Colonial militia surrounded the city of Boston.

An informal analysis of the event-based expressions in WikiWars suggested that the approach could be applied to only 34% of event-based expressions (44 out of 128);[40]

[40]By saying that the method could be applied we mean that it seemed to be possible to generate a search query that would not be too generic (for example, there might have been a number of sieges in Boston in the whole of history, so the query needs to be rather unambiguous), not that the method would necessarily provide a correct temporal value.

but in all but three of these cases, the correct reference time could also be found in the text.

Inevitably, many cases are more complex. Independently of whether we are using the content of the document or external resources like the web, in some cases coreference resolution must first be carried out. For example, to interpret *ten days after the battle began* we need to first establish what battle is meant; then we need to realise that we need the time when the battle began, not the whole timeframe of the battle.

6.5.7 Time Flow in Speech Transcripts

An issue that arises in processing temporal expressions in speech transcripts is the time flow and interpretation of deictic expressions. As noted earlier, for these expressions the reference time is the time-stamp of the utterance. However, if the speech act lasts for a considerable amount of time, then some expressions, e.g. *right now* or *two minutes ago*, may no longer be correctly interpreted using the time-stamp of the beginning of the act. This is a problem that occurs not only when processing the text automatically, but also when the text is being marked-up by human annotators. What we would really need is, for example, the time-stamp of each uttered sentence, or, in the case of a dialogue, each turn a speaker takes. This may or may not be important from a practical point of view, and it really depends on the application using the extracted temporal information.

6.5.8 Sentence-initial Temporal Adverbials

We also note that there has been some work done on some very specific issues related to the interpretation of temporal adverbials. Hitzeman (2005) claimed that, for example, there is a difference in the interpretation of *for an hour* in the following two sentences:

(6.57) a. Martha will be in her office *for an hour*.
b. *For an hour* Martha will be in her office.

According to Hitzeman, the first sentence has two possible readings: (1) Martha will be in her office for an hour in the not-precisely-specified future, and (2) Martha will be in the office for an hour starting at the time of the utterance. However, the second sentence, with the sentence-initial temporal adverbial, has only the second interpretation. Hitzeman also observed that sentence-initial adverbials are more common in narratives than in non-narratives and especially in narratives with many flashback scenes (as opposed to narratives with a simple story line). This is supposed to be so because authors of narratives avoid using ambiguous sentence-final temporal adverbials. We note that such a sophisticated analysis of the use of temporal expressions has been beyond the tasks defined in the information extraction community so far.

6.6 Conclusions

This chapter explored a number of issues involved in the interpretation of temporal expressions. In our approach, we begin by generating context-independent local semantic representations. These are then transformed into global semantic representations by taking into account the context of the text surrounding the expressions.

We first presented LTIMEX, our string-based representation of local semantics which is compatible with existing annotation schemes. This not only allows for more detailed evaluation of temporal expression taggers, but could also constitute an interface between different recognition and interpretation modules.

We then investigated the problem of finding the reference time for the interpretation of context-dependent expressions. While some of the descriptions of the taggers found in the literature briefly mention how the reference time is determined, usually they lack any evaluation of the chosen approach. We experimented with the various methods presented in the literature and proposed a variation that provided the best results on a combined dataset containing the WikiWars documents and a sample of 120 documents from the ACE 2005 Training corpus.

We then experimented with the interpretation of bare weekday names. The problem here is that apart from finding the reference time, it must be 'guessed' in which direction the referred-to weekday lies on the timeline in relation to the reference time. We provided the first comparative evaluations of the various methods discussed in the literature and proposed a modification to one of the methods that turned out to improve the accuracy.

Finally, we discussed a number of issues involved in the interpretation of various types of temporal expressions, indicating that development of a broad-coverage temporal expression tagger must deal with many specific details and cases, and is far from straightforward. All the issues presented here were observed during our analysis of real world documents.

Chapter 7

The DANTE System

One of the objectives of this thesis, and the ultimate goal of the research the thesis contains, is the development of a software system that can identify temporal expressions in text documents and express the meaning of these expressions in a chosen semantic representation. In this chapter we present the details of DANTE, a temporal expression tagging system that meets this objective.

DANTE is a rule-based system that consists of two main processing modules: a recognizer and an interpreter. The recognizer finds occurrences of temporal expressions in documents, determines their full extents in text, and analyses their local meaning to generate their LTIMEX values. Then, for each recognized temporal expression the interpreter determines its global semantic value. Depending on the taxonomical type of the expression the required further processing may be different; for example, for context-independent expressions the local and global values are the same; for all context-dependent temporal expressions a reference time must be found in the document; and for bare weekday names a direction of interpretation must be determined.

In Section 7.1 we introduce the system's architecture and some of the technical details, focusing on the two key components of the system. Then, in Section 7.2 we present the results of an evaluation of the system and discuss various aspects of the system's performance.

7.1 A Description of the System

In this section we present DANTE,[1] a system for the automatic tagging of temporal expressions in text documents. This is a rule-based system, developed as a general-purpose tool that can perform the TERN task as defined in the ACE program (NIST, 2007). Consequently, it aims to recognize temporal expressions of the types covered by the TIMEX2 annotation scheme, and to represent the meaning of these expressions using the semantics-related attributes defined in the annotation guidelines of the scheme (see (Ferro et al., 2005)).

Text processing in DANTE is organized into a pipeline of various processing resources run using the architectural constructs provided in the GATE platform (Cunningham et al., 2002); an overview of the pipeline's components is shown in Figure 7.1. We distinguish three major stages in the pipeline: **preprocessing**, **recognition** and **interpretation**. An input document is first preprocessed by a number of third-party tools that provide various types of annotations concerning the linguistic features of the text, such as the boundaries and parts-of-speech of words.[2] The input document is then processed by the DANTE Recognizer, which finds occurrences of temporal expressions and provides representations of their local semantics. The results of this recognition stage may be exported to an output document in those situations where the global semantics is to be derived by another tool. If the entire interpretation process is to be carried out by DANTE, then the text with the results provided by the recognizer is passed on in the pipeline to the DANTE Interpreter, which provides global semantics representations for the already-identified temporal expressions. We describe the processing of these parts of the pipeline in Sections 7.1.1–7.1.3.

The development of DANTE was carried out in a number of steps. In the first of these, a version of the system was developed on the basis of the TIMEX2 guidelines and the examples contained therein. Then we tested this version on the ACE 2005 training data and identified frequently-occurring cases which were problematic for the system. Addressing these problems constituted a second stage of system development. Subsequent development has focussed on fixing bugs and other similar errors discovered as we have applied the system to an ever-wider range of document data sources, including Australian Securities Exchange announcements,[3] the New York Times corpus[4] and the Reuters corpus.[5] We also experimented with syntax-based approaches to extent recognition, as discussed in Chapter 5. Finally, we improved the coverage of the system based on the expressions found in the WikiWars corpus, presented in Chapter 3.

[1]The name stands for Detection And Normalisation of Temporal Expressions.

[2]We acknowledge that in computer science the term 'preprocessing' is often used to refer to preliminary processing of a technical nature, such as a data format conversion, name normalisation, or abbreviation expansion. We use the term here in a much broader sense, referring to any processing that happens before applying the core components of DANTE; this may include quite complex tasks, such as named-entity recognition, which in some applications may constitute the main processing task.

[3]The Australian Securities Exchange (ASX) requires that registered companies publish various types of disclosures in circumstances that seem to be important for the financial market. The collection of the announcements is made available by SIRCA to its partners, including Macquarie University; see `http://www.sirca.org.au/display/SBX/Australian+Company+Reference+Data`. We worked with the subset containing the announcements from year 2006 and 2007.

[4]The corpus is distributed by LDC, see `http://www.ldc.upenn.edu/Catalog/catalogEntry.jsp?catalogId=LDC2008T19`.

[5]The corpus is distributed by NIST, see `http://trec.nist.gov/data/reuters/reuters.html`. A detailed description of the corpus can be found in (Lewis et al., 2004).

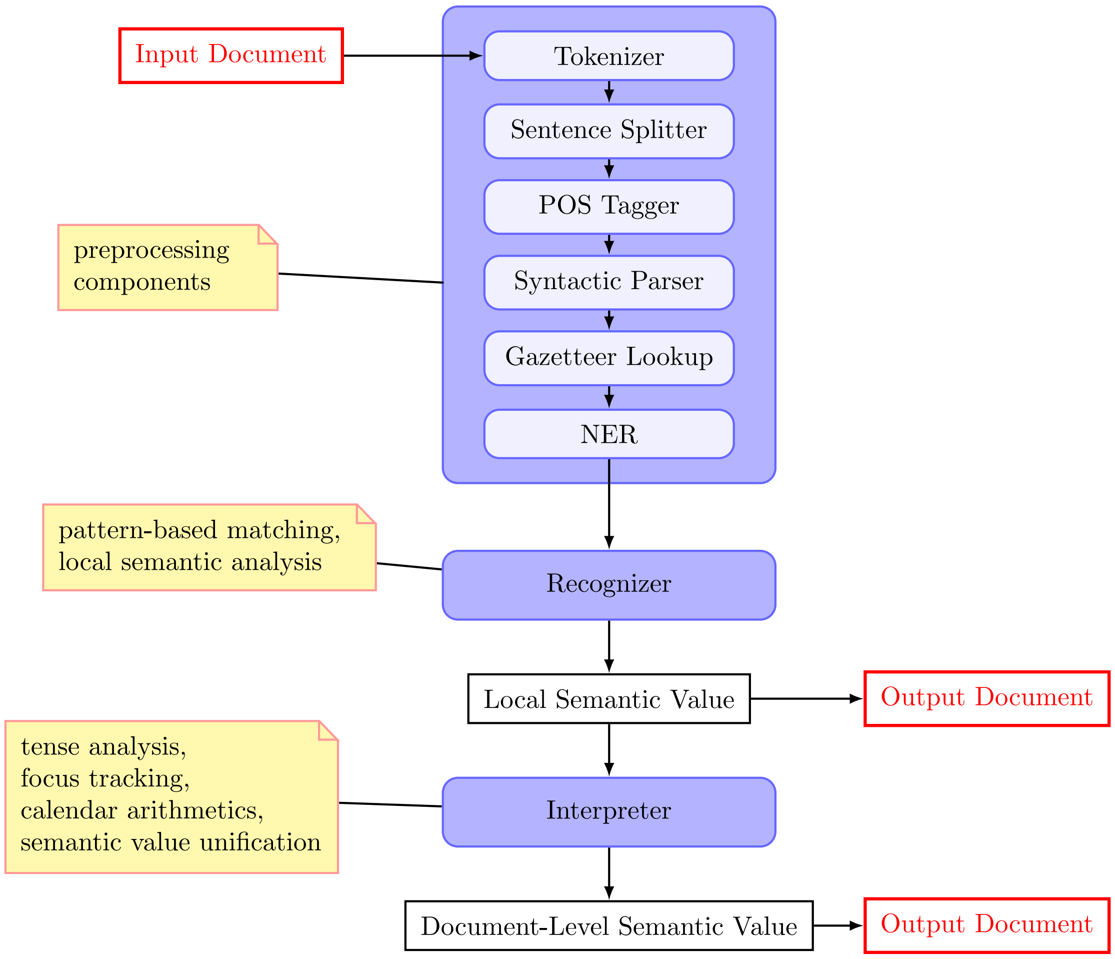


Figure 7.1: Pipeline-based processing in the DANTE system.

Independent of these extensions to the recognition grammar, we also carried out the experiments on weekday name interpretation and temporal focus tracking, as described in Chapter 6, integrating the best-performing methods into the system.

7.1.1 Preprocessing Components

The purpose of the preprocessing stage is to provide—by means of annotations that will be available to the subsequent processing steps—various kinds of information that can be useful for the recognition and interpretation of temporal expressions. The elements which can be used in this part of the pipeline are a tokenizer, a sentence splitter, a part-of-speech (POS) tagger, a syntactic parser, gazetteer lookup, and a named entity recogniser.

The tokenizer segments the input text into tokens, such as words, numbers and punctuation. These are then used to determine the locations of sentence boundaries, which are required, for example, for part-of-speech (POS) tagging and syntactic analysis. We use parts-of-speech in DANTE's recogniser to disambiguate the use of lexical items, allowing us to distinguish, for example, whether *second* is used as a noun, a verb or a number. POS tags are also used to check the tense of a sentence in the interpretation of temporal expressions. The gazetteer lookup performs a dictionary-based search for occurrences of specific tokens in the text, such as the names of temporal units, weekdays or months. A named-entity recognizer (NER) is included in the pipeline to facilitate the detection of addresses or people's names; these help us to distinguish temporal expressions from those expressions that have the same surface form but in fact are not temporal expressions (e.g. a four digit number being a part of an address does not denote a year, and *Day* or *Sun* may be the last name of a person rather than a temporal expression). Also, a syntactic parser may be used if the dependency-based or constituency-based approach is to be used for the recognition of temporal expressions (see Sections 5.4 and 5.5) or to find the controlling verb for a given temporal expression (see Section 6.4).

In the version of the system evaluated in Section 7.2, the preprocessing tools used are the rule-based components distributed as the ANNIE plugin to the GATE platform.[6] These are: the Default English Tokeniser, the RegEx Sentence Splitter, the reimplementation of Hepple's POS tagger, the ANNIE Gazetteer and the ANNIE Named Entity Recognizer. No syntactic parser is included in this pipeline because: i) we do not use the dependency-based and constituency-based approach to extent recognition since they perform worse than our pattern-based recognition grammar, and ii) we determine the controlling verb with heuristics based on part-of-speech tags.

7.1.2 The Recogniser

The temporal expression recognizer is based on a JAPE (Java Annotation Patterns Engine) grammar (Cunningham et al., 2000). JAPE is a component of the GATE platform which provides us with the ability to build a cascaded finite-state recognizer. Such a recognizer can consist of a number of transducers executed in sequence in a fashion similar to how the well-known FASTUS named entity recognizer worked (see (Appelt et al., 1995)). Each transducer contains rules which match annotations introduced to the document representation by the processing components appearing earlier in the pipeline (e.g. the annotations of tokens) and the preceding transducers of the grammar (e.g. annotations of partially recognized temporal expressions). Such a grammar is also referred to as a 'multiphase' grammar, with each transducer corresponding to one phase in the sequence.

DANTE's recognition grammar consists of seven phases. Each phase of the grammar processes the whole document from its beginning to its end. The initial phase contains only the expansion of macros used in the grammar rules of the remaining six phases. The second phase is the main one, in the sense that it is the biggest in terms of the number of rules (there are 244 rules) and that these rules detect the expressions in text. The rules in the following four phases modify the extent (most usually by including new surrounding tokens) and the representation of local semantics. One of these intermediate phases is entirely dedicated to modified expressions (e.g. *five months or more*). The penultimate phase introduces TIMEX2 annotations, since the previous phases operate on temporary annotation types (e.g. `TempDayPart`, `TempDayOfWeek`). The last phase is a post-processing phase in which we remove these intermediate annotations generated by the rules found in the intermediate phases (during debugging and development the post-processing phase is not included into the pipeline since we want to maintain these temporary annotations). The number of rules in the different phases varies; one of the phases has only one rule, while the largest part of the grammar has

[6]The ANNIE plugin is included in the distribution of GATE available at `http://gate.ac.uk`.

```
Rule: TimeAnalogueClock1     //   '5 in the morning'      '5 am'
(
 (HOUR_GAZ | ONE_DIGIT | TWO_DIGIT):hour
 (IN_THE_MORN_EVE | TIME_AMPM):ampm
):time
-->
{
  gate.AnnotationSet hourAnnSet = (gate.AnnotationSet)bindings.get("hour");
  String hour=doc.getContent().getContent(hourAnnSet.firstNode().getOffset(),
                              hourAnnSet.lastNode().getOffset()).toString();

  gate.AnnotationSet ampmAnnSet = (gate.AnnotationSet)bindings.get("ampm");
  String ampm=doc.getContent().getContent(ampmAnnSet.firstNode().getOffset(),
                              ampmAnnSet.lastNode().getOffset()).toString();

  String timeRepr = dante.HelpCalc.getReprForTime(hour,"00","","",ampm,"");

  gate.FeatureMap features = Factory.newFeatureMap();
  features.put("L-VAL", timeRepr);
  features.put("rule", "TimeAnalogueClock1");

  gate.AnnotationSet timeAnnSet = (gate.AnnotationSet)bindings.get("time");
  gate.Node firstNode = timeAnnSet.firstNode();
  gate.Node lastNode = timeAnnSet.lastNode();
  outputAS.add(firstNode,lastNode,"TempExTimeName",features);
}
```

Figure 7.2: An example JAPE rule in the DANTE's Recogniser module.

244 rules. Altogether the grammar consists of 98 macros and 343 rules.

JAPE rules are traditional pattern–action rules, where the left-hand side (LHS) contains the pattern to be matched, and the right-hand side (RHS) specifies the action to be taken when the pattern is matched. The pattern on the left-hand side is written using JAPE syntax, but the right-hand side can be written either in JAPE or directly in Java code; the latter provides much more functionality and flexibility. In Figure 7.2 we present an example rule called `TimeAnalogueClock1`;[7] it recognizes full hour times expressed using digits or words which are then followed by day part information (for example, *five in the morning* or *5 am*). The LHS, which is between the rule name and the `-->` separator, uses macros defined in the recognition grammar. The RHS is Java code; we first read the text matched by the sections of the LHS labeled `hour` and `ampm`, then we obtain the value representing the local semantics of the temporal expression, and finally we generate a `TempTimeName` annotation with two attributes `L-VAL` and `rule`.

Some of the rules in the grammar are 'negative'; they are used to recognize strings that are not temporal expressions, but which are similar in appearance to temporal expressions. These rules are included in the grammar to prevent other rules from matching expressions which are not time-referring. For example, there is a rule which

[7]Ideally, we would provide the listing of the whole grammar as an appendix to the thesis; however, the size of the code (approx. 15,000 lines) makes it impractical to do so.

```
Macro: TWO_DIGIT
({Token.kind == number, Token.length == "2"})

Macro: IN_THE_MORN_EVE
(
 ( {Token.string == "in"}  THE )?
 ( {Token.string == "morning"} | {Token.string == "afternoon"} |
   {Token.string == "evening"}
 )
)
```

Figure 7.3: Example JAPE macros in DANTE's Recogniser module.

matches *March* as a month name; but we also have a negative rule that finds this word in contexts where it is not used to denote a month name, as in *March on Vilno*; this prevents the generation of spurious annotations in such cases.

The macro expansions are textually copied into the bodies of rules by the JAPE engine, and then the rules are compiled into Java code. The content of a macro therefore has the same syntax as the LHS of a rule. Figure 7.3 presents two examples of macros that are used in rule `TimeAnalogueClock1`. They match those annotations of text tokens that meet the specified requirements, for example the kind or length of the token, or the actual string that makes up the word.

The recognition rules and macros use 33 gazetteers with a total of about 1200 entries: these are strings used in the expression of dates and times, such as numbers written in words or the names of weekdays, months, year seasons and time zones. This apparently large total number of entries is due to two factors: variations in names and their spellings (e.g. *All Saints'*, *All-Saints'*, *All Saints' Day*, *All-Saints' Day*); and the case sensitivity of the lookup mechanism, which requires us to list all the casing variations we might expect to appear—for example, as an abbreviated variant of *Thursday* we have entries for *Thu*, *THU*, and *thu*. Also, some categories of entries are simply numerous: for example, there are 39 time zones and many can be referred to with a number of names—the -05:00 time zone can be referred to as *Eastern Standard Time*, *EST*, *Eastern Time*, *Romeo Time*, or *RTZ*. In consequence, we have 269 entries corresponding to the names of time zones alone.

Since the recognizer also generates the representation of local semantics, the module is integrated with a library for the normalisation of values. This lets us easily convert in the RHS of a rule, for example, a month name used in an expression to a numeric value to be used in the LTIMEX representation, independently of the specific surface form that was used in the text. For example, *September*, *Sept*, *Sep.* and *9* (in a context where this refers to a month) are converted to `09` in the value of the `L-VAL` attribute. Similarly, the recognizer can obtain a value for some named dates, such as *Christmas Day* being represented as `xxxx-12-25`.

Also integrated into the recognizer are the verification of basic calendar constraints and some calendar-oriented operations, which allow us to check whether a recognized string can denote a valid date. For example, knowledge of the number of days in each month allows us to determine that the matched string *2010/02/31* is unlikely to be a temporal expression. This library also provides us with algorithms for the calculation of the dates of some named days, such as Ash Wednesday and Easter Sunday; in contrast to Christmas Day, the exact dates of these days depend on the year.

7.1.3 The Interpreter

The interpreter module processes a document sentence by sentence. Each temporal expression identified by the recognizer is now analysed by the interpretation module, which transforms the local semantic representation into a document-level (i.e. global) semantic representation which is expressed by means of the five TIMEX2 attributes: `VAL`, `ANCHOR_VAL`, `ANCHOR_DIR`, `MOD`, and `SET`.

The VAL attribute Depending on the type of the temporal expression being interpreted (e.g. explicit, underspecified, offset, duration), different interpreting actions are taken. For context-independent expressions, the value of the `VAL` attribute is the same as the value of `L-VAL`. For context-dependent expressions, two basic operations are used: (1) unification of an underspecified semantic value with some reference value, and (2) the addition/subtraction of a specified number of units to/from a reference value. These are carried out by a calendar arithmetic library available to the interpreter. Both of these operations require a reference time value. In our current model, we assume that a document has a simple linear structure, and that any hierarchical structure in the document has no bearing on the interpretation of temporal expressions. Given these assumptions, the reference time is selected using the recency-based approach (see Section 6.3), using the most recent point expression of sufficiently fine granularity. For some document types, such as news, it is also safe to make the simplifying assumption that the temporal focus used to compute document-level values for temporal expressions does not advance during the processing of the document, so that the time-stamp of the document can be used to interpret all context-dependent expressions. Both assumptions may not always hold true, but are very likely to work for the majority of cases, as we demonstrated in Section 6.3. For the final evaluations of DANTE presented in Section 7.2 we processed the ACE corpora with the document time-stamp approach, and the WikiWars corpus with the 'most recent point expression' approach.

The calendar arithmetic library also provides additional functionalities for checking which half-year or quarter-year a given date falls within; checking whether a year is a leap year; calculating the ISO week number for a given date; converting a date from month-based format (`yyyy-mm-dd`) to week-based format (`yyyy-Wnn-m`) and vice versa; checking the weekday of a date; calculating the date of an expression which refers to the nth weekday name in a month (e.g. *the second Tuesday of June*); and calculating the difference in days between two dates.

The interpreter also implements the approaches for determining the direction of interpretation of bare weekday names presented in Section 6.4.

The MOD attribute The `MOD` attribute encodes information about semantic modifications that apply to the value represented in the `VAL` attribute. Since these modifications are realised in the text by some of the tokens that are included in the extent of a temporal expression, the target value of the `MOD` attribute is already provided in the `L-MOD` attribute by the recognizer. Therefore, the task for the interpreter is simply to replicate this value.

The ANCHOR_VAL and ANCHOR_DIR attributes The ANCHOR_VAL and ANCHOR_DIR attributes are used with anchored periods and general references to the past, present and future. As we indicated in Section 6.5, finding the values of these attributes is generally a challenging problem. Currently, we use the document timestamp to provide the value of ANCHOR_VAL; and ANCHOR_DIR is filled with the value of L-ANCHOR_DIR, which is determined by the recognizer based on the lexical items found within the extent of the temporal expression. A more sophisticated approach could analyse the text to determine the value of the anchor and to decide whether it is the starting or ending point of a period.

The SET attribute As in the case of the MOD attribute, the value of the SET attribute (which represents a binary choice as to whether the expression is or is not a reference to a set) replicates the decision made by the recognizer which filled in the L-SET attribute. The estimation of whether an expression refers to a single temporal entity or to a set of such entities in most cases can be made without access to the context.[8]

7.2 Evaluation

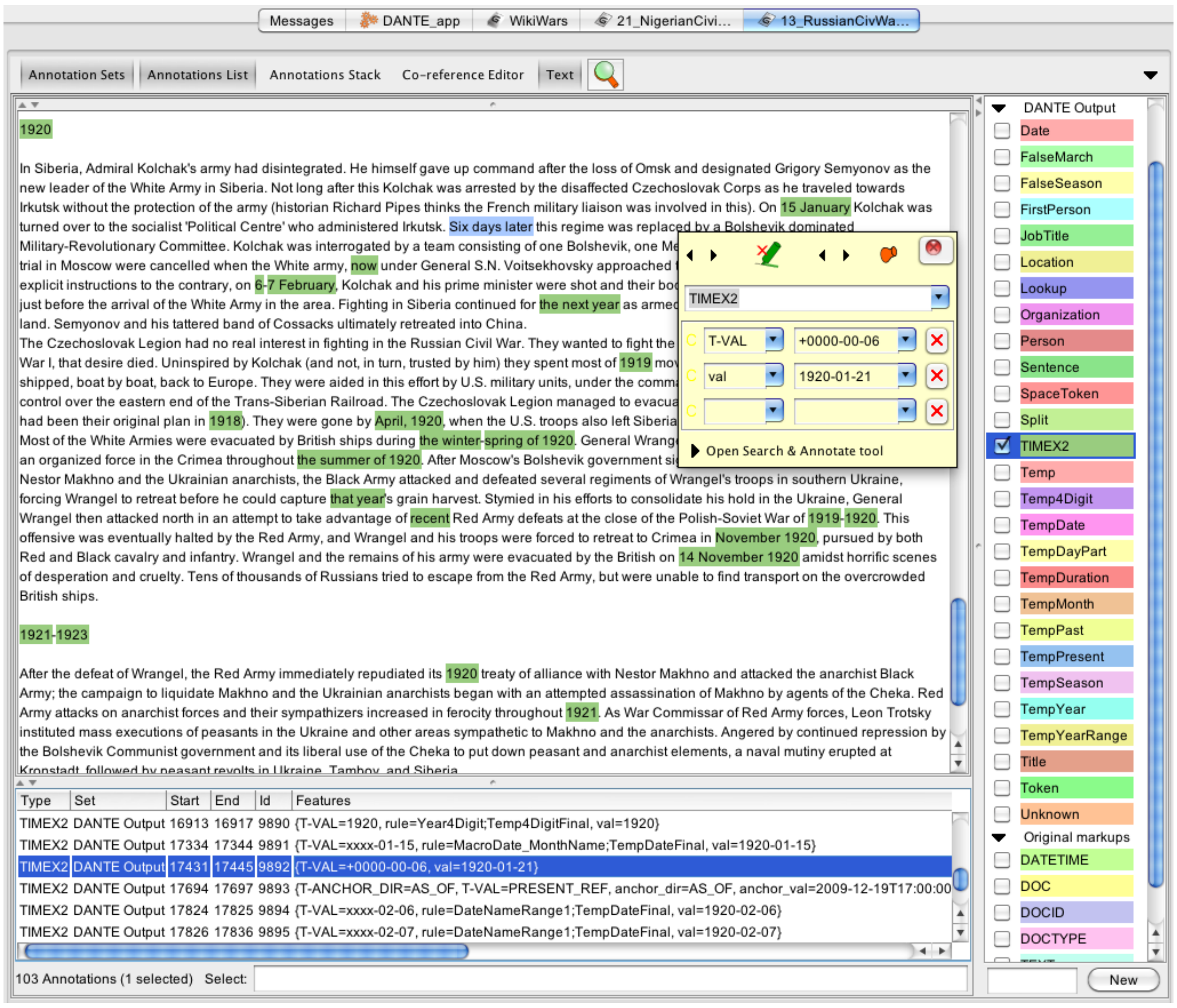
The standard approach to the evaluation of an information extraction system is to run the system on benchmark datasets and compare the performance with the score an ideal system would get; the annotations in such an evaluation corpus are often referred to as the **gold standard**. Of course, in order for the results to be valuable, the benchmark data must represent a relatively broad range of phenomena and problems for which the system was developed. The advantage of using standard benchmark data is the possibility of comparing a given system's performance to the performance of other systems evaluated on the same datasets.

To evaluate the performance of DANTE, we use the Corpus Quality Assurance (CQA) tool available in GATE and the ACE 2007 evaluation script. The former provides us with the typical measures—precision, recall and F-measure—both for the evaluation of extent recognition only and also combined with the evaluation of selected semantic attributes; the latter provides us with the ACE Value, a measure used at the official ACE evaluations. These metrics are discussed in more detail in Appendix C.

Since DANTE is developed as a plugin to GATE, we can easily use the CQA tool when running the DANTE pipeline in the GATE Developer; for the use of the ACE evaluation script we built a standalone application that runs DANTE's pipeline and exports results to versions of the input texts with inline annotations, which are then converted to the ACE APF format with the tool prepared by NIST for the official ACE 2004 evaluations.

We also use the document editor available in the GATE Developer; this allows visual inspection of system performance by highlighting the annotations generated by the system and those provided in the gold standard. Figure 7.4 shows a view of the tool displaying a document with the temporal expressions found by DANTE.

[8]The cases where context is required include certain generic expressions, as in '*April* is usually wet'.



1920

In Siberia, Admiral Kolchak's army had disintegrated. He himself gave up command after the loss of Omsk and designated Grigory Semyonov as the new leader of the White Army in Siberia. Not long after this Kolchak was arrested by the disaffected Czechoslovak Corps as he traveled towards Irkutsk without the protection of the army (historian Richard Pipes thinks the French military liaison was involved in this). On 15 January Kolchak was turned over to the socialist 'Political Centre' who administered Irkutsk. Six days later this regime was replaced by a Bolshevik dominated Military-Revolutionary Committee. Kolchak was interrogated by a team consisting of one Bolshevik, one Me
trial in Moscow were cancelled when the White army, now under General S.N. Voitsekhovsky approached
explicit instructions to the contrary, on 6-7 February, Kolchak and his prime minister were shot and their bo
just before the arrival of the White Army in the area. Fighting in Siberia continued for the next year as armed
land. Semyonov and his tattered band of Cossacks ultimately retreated into China.
The Czechoslovak Legion had no real interest in fighting in the Russian Civil War. They wanted to fight the
War I, that desire died. Uninspired by Kolchak (and not, in turn, trusted by him) they spent most of 1919 mov
shipped, boat by boat, back to Europe. They were aided in this effort by U.S. military units, under the comma
control over the eastern end of the Trans-Siberian Railroad. The Czechoslovak Legion managed to evacua
had been their original plan in 1918). They were gone by April, 1920, when the U.S. troops also left Siberia
Most of the White Armies were evacuated by British ships during the winter-spring of 1920. General Wrange
an organized force in the Crimea throughout the summer of 1920. After Moscow's Bolshevik government si
Nestor Makhno and the Ukrainian anarchists, the Black Army attacked and defeated several regiments of Wrangel's troops in southern Ukraine, forcing Wrangel to retreat before he could capture that year's grain harvest. Stymied in his efforts to consolidate his hold in the Ukraine, General Wrangel then attacked north in an attempt to take advantage of recent Red Army defeats at the close of the Polish-Soviet War of 1919-1920. This offensive was eventually halted by the Red Army, and Wrangel and his troops were forced to retreat to Crimea in November 1920, pursued by both Red and Black cavalry and infantry. Wrangel and the remains of his army were evacuated by the British on 14 November 1920 amidst horrific scenes of desperation and cruelty. Tens of thousands of Russians tried to escape from the Red Army, but were unable to find transport on the overcrowded British ships.

1921-1923

After the defeat of Wrangel, the Red Army immediately repudiated its 1920 treaty of alliance with Nestor Makhno and attacked the anarchist Black Army; the campaign to liquidate Makhno and the Ukrainian anarchists began with an attempted assassination of Makhno by agents of the Cheka. Red Army attacks on anarchist forces and their sympathizers increased in ferocity throughout 1921. As War Commissar of Red Army forces, Leon Trotsky instituted mass executions of peasants in the Ukraine and other areas sympathetic to Makhno and the anarchists. Angered by continued repression by the Bolshevik Communist government and its liberal use of the Cheka to put down peasant and anarchist elements, a naval mutiny erupted at Kronstadt, followed by peasant revolts in Ukraine, Tambov, and Siberia

```
Type   Set          Start End   Id   Features
TIMEX2 DANTE Output 16913 16917 9890 {T-VAL=1920, rule=Year4Digit;Temp4DigitFinal, val=1920}
TIMEX2 DANTE Output 17334 17344 9891 {T-VAL=xxxx-01-15, rule=MacroDate_MonthName;TempDateFinal, val=1920-01-15}
TIMEX2 DANTE Output 17431 17445 9892 {T-VAL=+0000-00-06, val=1920-01-21}
TIMEX2 DANTE Output 17694 17697 9893 {T-ANCHOR_DIR=AS_OF, T-VAL=PRESENT_REF, anchor_dir=AS_OF, anchor_val=2009-12-19T17:00:00
TIMEX2 DANTE Output 17824 17825 9894 {T-VAL=xxxx-02-06, rule=DateNameRange1;TempDateFinal, val=1920-02-06}
TIMEX2 DANTE Output 17826 17836 9895 {T-VAL=xxxx-02-07, rule=DateNameRange1;TempDateFinal, val=1920-02-07}
103 Annotations (1 selected)  Select:
```

Figure 7.4: A view of GATE Developer with a document processed using DANTE.

7.2.1 Evaluation on Gold-Standard Datasets

With respect to the processing of temporal expressions, the most widely-recognized evaluations in the community are the ACE evaluations. We participated in the ACE 2007 evaluations, using an early version of DANTE for the English TERN task. This provided us with a chance to compare our results to those obtained by others in this area. Also, as a result of the participation, two gold-standard corpora were made available to us, the ACE 2005 Training corpus and the ACE 2007 Evaluation corpus, which we use for the evaluation of the current version of the DANTE system. Details of these corpora are presented in Section 2.5; here, we only briefly review some basic facts about them.

The ACE 2005 Training corpus contains 593 documents with 5,428 annotations of temporal expressions; the documents are organized into six domains (broadcast conversations, broadcast news, newswire, telephone conversations, UseNet discussions and web blog entries) and each document can belong to only one domain. The domains are unequally sized, both in terms of the number of documents and the number of temporal expressions they contain; see Table 7.1. The ACE 2007 Evaluation corpus is quite similar: it is organized into the same six domains as the ACE 2005 Training corpus, but it is smaller; it contains 254 documents with 2,028 annotations of temporal

Domain	Docs	Data size [b]	TIMEX2
BC	59	260,413	618
BN	225	378,874	1,450
CTS	39	299,139	409
NW	106	333,328	1,235
UN	49	235,415	741
WL	115	249,251	975
Total	593	1,756,420	5,428

Table 7.1: Statistics for the ACE 2005 Training corpus.

Domain	Docs	Data size [b]	TIMEX2
BC	9	48,722	142
BN	74	75,731	322
CTS	6	54,522	70
NW	106	209,973	894
UN	13	48,377	167
WL	46	137,549	433
Total	254	574,874	2,028

Table 7.2: Statistics for the ACE 2007 Evaluation corpus.

Document ID	Tokens	TIMEX2
01_WW2	5,593	170
02_WW1	10,370	265
03_AmCivWar	3,529	75
04_AmRevWar	5,695	147
05_VietnamWar	11,640	245
06_KoreanWar	5,992	149
07_IraqWar	8,404	247
08_FrenchRev	9,631	175
09_GrecoPersian	7,393	129
10_PunicWars	3,475	57
11_ChineseCivWar	3,905	102
12_IranIraq	4,508	98
13_RussianCivWar	3,924	104
14_FirstIndochinaWar	3,085	71
15_MexicanRev	3,910	78
16_SpanishCivilWar	1,455	63
17_AlgerianWar	7,716	130
18_SovietsInAfghanistan	5,306	110
19_RussoJap	2,760	62
20_PolishSoviet	5,137	106
21_NigerianCivilWar	2,091	29
22_SecondItaloAbyssinianWar	3,949	69
Total	119,468	2,681

Table 7.3: Statistics for the WikiWars corpus.

expressions (see Table 7.2). Note that the ACE 2007 Evaluation corpus was not used for the development of DANTE, however, so that we would have unseen data for testing.

We also use the WikiWars corpus since it provides the quite different domain of narratives about protracted geo-political events. Although it contains only 22 documents, it has 2,681 annotated temporal expressions. The exact distribution of the annotations across the documents is presented in Table 7.3.[9]

7.2.1.1 Expression Detection and Extent Recognition

With respect to evaluation of the identification of temporal expressions in texts, there are two tasks to be considered: expression detection and extent recognition. In the former case we require only some overlap between the system-produced annotation and the corresponding gold-standard annotation; in the latter case we require the extent of the system-produced annotation to be identical to the extent of the gold-standard annotation.

The Corpus Quality Assurance tool considers a gold-standard annotation to be detected by the system if at least one of its characters is annotated by the system.

[9]Tables 7.1, 7.2 and 7.3 are repeated here from Table 2.13 and Table 2.14 on page 48, and Table 3.3 on page 81, to avoid the reader needing to have to locate them earlier in the text.

					Strict			Lenient		
Corpus	Corr.	Miss.	Spur.	Part.	Prec.	Recall	F	Prec.	Recall	F
ACE 2005 Train.	4350	456	557	622	0.79	0.80	0.79	0.90	0.92	0.91
ACE 2007 Eval.	1582	199	157	247	0.80	0.78	0.79	0.92	0.90	0.91
WikiWars	2484	41	44	156	0.93	0.93	0.93	0.98	0.98	0.98

Table 7.4: The evaluation results for expression detection and extent recognition using DANTE.

			Strict			Lenient		
System	Reference	Corpus	Prec.	Recall	F	Prec.	Recall	F
TagTime	Baldwin (2002)	TIDES part 1	0.60	0.47	0.53	–	–	–
TagTime	Baldwin (2002)	TIDES part 2	0.65	0.50	0.56	–	–	–
Chronos	Negri and Marseglia (2005)	ACE'04 Eval.	0.89	0.80	0.84	0.98	0.88	0.93
BRO	cited by Ahn et al. (2007)	ACE'04 Eval.	0.91	0.83	0.87	0.97	0.89	0.93
ATEL	Hacioglu et al. (2005)	ACE'04 Eval.	0.92	0.84	0.88	0.98	0.89	0.94
–	Ahn et al. (2005a)	ACE'04 Eval.	0.86	0.75	0.80	0.98	0.86	0.91
–	Ahn et al. (2007)	ACE'04 Eval.	0.88	0.77	0.82	0.93	0.81	0.87
HeidelTime	Strötgen and Gertz (2010)	WikiWars	0.86	0.75	0.80	0.94	0.82	0.88

Table 7.5: The evaluation results for expression detection and extent recognition obtained by other systems reported in the literature; BRO stands for Best Recognition Only and refers to the best-performing system at ACE 2004 Evaluations which carried only the recognition task (no interpretation).

In the ACE evaluations, an annotation generated by a system is classified as matched with an annotation from the gold standard if there is minimum 30% text span overlap between them.

Table 7.4 presents the results for extent recognition measured on the three datasets we used. The strict results concern precise extent recognition, and lenient results concern the detection task.

With only some small differences in precision and recall, both the strict and lenient F-measures are the same for the two ACE corpora, at 0.79 and 0.91 respectively. The results for WikiWars are higher; the F-measures are 0.93 and 0.98. Given these high results for both the detection and recognition tasks across the diversity of domains found in our datasets, we believe it is appropriate to consider DANTE to be broad-coverage.

And although the system is not perfect and there is still room for improvement, in the areas of both recognition and interpretation, the results are satisfactorily high and can be considered state-of-the-art. In Table 7.5 we present those results reported in the literature that can be—to a limited degree—considered alongside ours because they use the same evaluation metrics. Unfortunately, most of these results cannot be compared directly to ours because different evaluation data has been used; furthermore, these were annotated with a slightly different version of the TIMEX2 scheme. Nevertheless, they may still serve to give an idea of the current state-of-the-art.

We draw attention to two rows in Table 7.5 which represent the two best-performing systems submitted to the ACE 2004 evaluations. BRO is not the real name of a participating system, but rather is an acronym for an anonymous system which turned out to be the Best-performing Recognition Only system; Chronos was the best performing system that carried out the full TERN task (i.e. both the recognition and interpretation subtasks). For this reason, both BRO and Chronos can be considered state-of-the-art systems. In terms of the *raw* numbers, our results are worse by about 0.02–0.03 in F-measures for the detection task; for the recognition task, the difference in the F-measures is greater at about 0.05–0.09. Our results are at the level of those obtained by Ahn et al. (2005a) and Ahn et al. (2007). However, we cannot tell whether the differences are due to the performance of the systems, or to different experimental set-ups. Also, we do not know to what extent the authors tuned their systems to the ACE corpora; our goal was to stay maximally conformed with the TIMEX2 guidelines, but, as we show in the error analysis below (see Section 7.2.2), the gold-standard annotations in the ACE corpora do not always follow the guidelines precisely.[10]

In particular, we compare our results to the results obtained with the recently-developed HeidelTime tagger (Strötgen and Gertz, 2010), since it has been evaluated on our WikiWars corpus.[11] Here, our results are better by 0.10 in F-measure for the detection task, and by 0.13 in F-measure for the recognition task.[12] Based on this comparison, we argue that the performance of DANTE exceeds the performance of HeidelTime.

In Table 7.9 we also provide the F-measures for the detection and recognition tasks for each individual document in the WikiWars corpus. The strict F-measure ranges from 0.86 (for 19_RussoJap) to 0.99 (for 13_RussianCivWar); the lowest lenient F-measure is 0.97 and the highest is 1.00.

7.2.1.2 Extent Recognition and Interpretation of Meaning

With the evaluation tools we use, we have two means to measure the quality of the interpretation process. With the Corpus Quality Assurance we can calculate precision, recall and F-measure for the extent recognition and values assigned to selected attributes; for the annotation to be treated as correctly recognized both the precise extent and the attribute value must be correct; if the extent is incorrect (but there is some overlap) or the attribute value is wrong then the annotation is considered to be 'partially correct'. The other possibility is to calculate the ACE Value as was done at the official ACE evaluations.

Results of the first kind are presented in Table 7.6; in the first row we repeat the results of extent recognition, and in the following rows we show the results combining the extent and a selected attribute. In the last row, we evaluate extent with all attributes, thus providing an evaluation of all aspects of the annotations. By comparing those results to those found in the first row, we can see what drop in performance is caused by incorrect generation of values for the TIMEX2 attributes.

[10]Of course, the taggers based on machine learning, e.g. BRO and ATEL, are prepared by using training data, rather than the annotation guidelines, and this way these taggers are naturally bound to be well-adjusted to any systematic deviance present in the annotated data.

[11]The results can be found at `http://dbs.ifi.uni-heidelberg.de/index.php?id=139` (last accessed on 22/11/2011).

[12]We note that HeidelTime was evaluated on the initial version of WikiWars (v1.0.0) while we use the most recent version 1.0.4, which has some annotation errors corrected (all the results for WikiWars in this thesis concern this later release of the corpus). However, when we evaluate DANTE on WikiWars v1.0.0, our results are even higher: the strict and lenient F-measures are 0.95 and 0.99, respectively.

	ACE 2005 Train.		ACE 2007 Eval.		WikiWars	
Attribute	Strict F	Lenient F	Strict F	Lenient F	Strict F	Lenient F
no attribute	0.79	0.91	0.79	0.91	0.93	0.98
VAL	0.61	0.66	0.68	0.73	0.81	0.84
MOD	0.77	0.87	0.78	0.89	0.92	0.98
ANCHOR_VAL	0.64	0.70	0.66	0.74	0.84	0.88
ANCHOR_DIR	0.67	0.74	0.69	0.78	0.90	0.94
SET	0.78	0.90	0.78	0.90	0.92	0.98
all attributes	0.51	0.53	0.56	0.60	0.74	0.75

Table 7.6: The evaluation results for DANTE's extent recognition and generation of values for TIMEX2 attributes.

	– ACE 2004 Eval. Ahn et al. (2005a)		HeidelTime WikiWars Strötgen and Gertz (2010)	
Attribute	Strict F	Lenient F	Strict F	Lenient F
no attribute	0.80	0.91	0.80	0.88
VAL	–	0.61	0.74	0.79
ANCHOR_VAL	–	0.44	–	–
ANCHOR_DIR	–	0.44	–	–

Table 7.7: The evaluation results for extent recognition and the generation of values for TIMEX2 attributes by other systems reported in the literature.

When comparing the results across the corpora used for evaluations, we note that for all the attributes considered individually and together, the best results are obtained for WikiWars, and the worst for the ACE 2005 Training corpus. That WikiWars provides the best performance is not surprising given the much higher results obtained here for extent recognition only; but it is interesting to note the difference between the two ACE corpora. Although the results for the detection and recognition tasks are the same, in the case of the interpretation task all the results are higher for the ACE 2007 Evaluation corpus than for the ACE 2005 Training corpus. This is not what we would expect, given that we used the 2005 corpus for the development of DANTE, and the 2007 corpus was not used for this purpose at all.

The most important attribute in TIMEX2 is `VAL`; not only this is the most often-used attribute, but it also carries the major part of the meaning representation. It is also the most weighted attribute in the ACE evaluations (see Table C.1 in Appendix C). Irrespective of the corpus used, the results for this attribute are the lowest of all the attributes. The results for `ANCHOR_VAL` are not much higher than in the case of `VAL`; however, since `ANCHOR_VAL` is less frequently used than `VAL`, it means that there are proportionally more incorrect values of `ANCHOR_VAL` than of the `VAL` attribute. Although `ANCHOR_VAL` is always used together with `ANCHOR_DIR`, the latter turns out to be less problematic as the results are higher. This is related to the fact that `ANCHOR_VAL` stores a date that must be calculated by the interpreter, but the value of `ANCHOR_DIR` is one of the values from a set predefined by the annotation guidelines. The results for the `MOD` and `SET` attributes are the highest and are very close to those obtained for extent recognition. There are two reasons for this. One is that relatively often these attributes

	ACE 2005 Training Corpus	ACE 2007 Evaluation Corpus				
			Official ACE 2007 Evaluations			
Domain	DANTE	DANTE	SystemA	SystemB	SystemC	SystemD (early DANTE)
Broadcast Conv.	60.2	55.5	44.2	48.2	46.6	30.0
Broadcast News	57.8	73.3	68.4	68.6	67.8	44.4
Newswire	70.4	70.2	67.4	60.9	57.3	54.2
Telephone Conv.	54.1	55.0	52.6	60.2	64.2	38.7
UseNet	58.3	70.6	63.1	58.2	59.0	55.9
Weblogs	64.5	63.3	51.4	52.9	54.8	44.8
Total	61.9	67.6	61.6	59.3	58.2	48.3

Table 7.8: The results of the DANTE system on the ACE 2005 Training and ACE 2007 Evaluation datasets.

have an empty value, so not generating their values in most cases already provides good results. Secondly, the class of expressions for which these attributes should be used and the values to be provided are well-defined, so it is relatively easy to produce the correct values.

When all attributes are evaluated, the results drop by 0.19–0.28 in strict F-measure, and by 0.23–0.38 in lenient F-measure compared to those for extent recognition. This may seem like a lot, and is definitely a reason to continue further development of DANTE; however, these results are still higher than those reported in the literature, which we present in Table 7.7.[13] Compared to the performance of HeidelTime evaluated on WikiWars, our results for the `VAL` attribute are higher by 0.05 for lenient F-measure and by 0.07 for strict F-measure.[14] While Ahn et al. (2005a) achieved practically the same performance for extent recognition as we did, their results for `VAL`, `ANCHOR_VAL`, `ANCHOR_DIR` are much lower than ours. Although we use a different ACE corpus than they did, the degree of difference here suggests that DANTE has better interpretation performance than their tagger. Depending on which ACE corpus we compare to, our lenient F-measures are better by 0.05–0.12 (`VAL`), 0.26–0.30 (`ANCHOR_VAL`) and 0.30–0.34 (`ANCHOR_DIR`).

The second possibility for measuring the performance of attribute value generation is to use the ACE Value. To calculate this measure we use the official scoring script used at the ACE 2007 evaluations. The results for the ACE corpora are presented in Table 7.8 and those for WikiWars are shown in Table 7.9.

[13]We note that some of the similar-looking evaluations of attribute values reported in the literature, for example by Negri and Marseglia (2005), are calculated not for all temporal expressions, as ours are, but only for those whose gold-standard annotations have the specific attribute filled-in. Since this may heavily impact the results and make them completely uncomparable, we do not provide them in Table 7.7. Both approaches have their own benefits; ours reflects the significance of the attribute (i.e. how many expressions it concerns), while the approach used by Negri and Marseglia (2005) really focuses on the evaluation of the quality of the attribute value generation.

[14]As we already noted, Strötgen and Gertz (2010) used a different version of the WikiWars corpus; however, when we use the same version as they did, our results are even higher. For example, for the `VAL` attribute the string F-measure would be 0.83 and the lenient F-measure would be 0.85.

Document ID	ACE Value	Extent	
		Strict F	Lenient F
01_WW2	83.0	0.95	0.99
02_WW1	76.6	0.90	0.99
03_AmCivWar	89.0	0.93	0.98
04_AmRevWar	87.7	0.95	0.99
05_VietnamWar	77.3	0.94	0.98
06_KoreanWar	80.8	0.92	0.97
07_IraqWar	81.1	0.93	0.99
08_FrenchRev	78.2	0.90	0.98
09_GrecoPersian	70.3	0.90	0.97
10_PunicWars	81.7	0.90	0.99
11_ChineseCivWar	86.7	0.91	0.97
12_IranIraq	88.3	0.94	0.98
13_RussianCivWar	89.9	0.99	1.00
14_FirstIndochinaWar	86.3	0.92	0.98
15_MexicanRev	88.3	0.94	0.97
16_SpanishCivilWar	84.6	0.91	0.99
17_AlgerianWar	80.1	0.89	1.00
18_SovietsInAfghanistan	83.1	0.93	0.98
19_RussoJap	76.0	0.86	0.99
20_PolishSoviet	74.8	0.98	1.00
21_NigerianCivilWar	96.0	0.97	1.00
22_SecondItaloAbyssinianWar	68.1	0.90	0.97
Total	81.0	0.93	0.98

Table 7.9: The performance of DANTE on the WikiWars documents.

As in the case of the analysis of F-measures, we notice that our results for the ACE 2005 corpus are lower than those for the 2007 corpus: 61.9 as opposed to 67.6. For some domains the results for the two corpora are very close: newswire (70.4 vs 70.2), telephone conversations (54.1 vs 55.0), and weblogs (64.5 vs 63.3). The most problematic domain turns out to be transcripts of telephone conversations; this is also the smallest domain in the two corpora. Considering both corpora, the top performing domain is newswire; for the ACE 2007 corpus, other two domains for which DANTE performed best were broadcast news (73.3) and UseNet discussions (70.6).

In Table 7.8 we also compare the results we obtain for the ACE 2007 corpus with the results achieved by the systems participating in the official ACE 2007 evaluations,[15] including our early version of DANTE. We are pleased to report that the current version of DANTE performs best of all these systems; moreover, for all domains except telephone conversations, DANTE provides the best results of all the systems. We note that, at the time of the official evaluations in 2007, each of the other ACE 2007 participants obtained the best results for two domains; this may suggest that the systems were optimized for some domains which, however, had the consequence of deteriorating their performance on the remaining four domains.

[15]See `http://www.itl.nist.gov/iad/mig/tests/ace/2007/doc/ace07_eval_official_results_20070402.html` (last accessed on 22/11/2011).

The overall ACE Value for WikiWars is 81.0. Given the low number of the documents in this corpus, in Table 7.9 we also provide the results for individual documents. The lowest ACE Value is 68.1 (for 22_SecondItaloAbyssinianWar) and the highest is 96.0 (for 21_NigerianCivilWar).

7.2.2 Error Analysis

Although the DANTE system performs very well, it still makes errors both in the recognition task and in the interpretation task. We can divide the erroneously generated annotations into three groups: (1) cases that can be fixed with more time spent on development, carrying out a fairly standard debugging exercise; (2) cases that are rather difficult to process correctly without the development of a more sophisticated approach to solving the problem; and (3) cases that result from errors in the gold-standard annotations of the corpora, thus requiring no action in terms of the development of the system, but pointing to a need to reflect on how to interpret the obtained evaluation results.

7.2.2.1 Cases for Debugging

Our error analysis showed that there are a number of cases where the recognition grammar needs more attention. One type of scenario is where we did not cater for a specific pattern. For example, in *a short time ago* we missed the last word (*ago*), resulting in both an incorrect extent and a wrong interpretation.

Other errors are due to the grammar not including all possible lexical items in a rule; for example, we miss a few expressions triggered by the word *weekend*.

Sometimes the rules do not work as we had intended because of interactions between the rules. For example, instead of recognizing the date in the conventionalised format *11/12/04*, a rule matching the turn of the year is activated (originally designed to recognize strings like *1980/1981* or *1980/81*) resulting in an incorrect recognition by annotating only *11/12*; of course the derived value of the expression is wrong too.

Also, in some cases we fail to combine partial matches into a single annotation. Such cases lower the results significantly, because they generate many errors. For example, if the system annotates *April next year* as two expressions, *April* and *next year*, it is heavily penalized, because such annotations result in one match with incorrect extent and most likely incorrect value (*April*), and one spurious annotation (*next year*). However, for the calculation of the ACE Value spurious annotations are very harmful, since not only is their occurrence (i.e. extent) penalized, but also each of their attributes provides negative points.

More attention must also be paid to expressions containing hyphenated words, such as *spring-time*, *mid-morning* or *four-year*; currently, because of technical issues related to the identification of tokens and gazetteer entries, some such expressions are missed or recognized only partially.

In terms of improvements of the ACE Value, the most significant impact would be achieved by addressing the issues related to the extent and the `VAL` and `ANCHOR_VAL` attributes. This is not only because of the high weights assigned to these features, but also because, as we showed in Table 7.6, these two attributes are the ones which most often receive an incorrect value.

7.2.2.2 Difficult Cases

We consider difficult cases to be those expressions for which providing a correct annotation is not just a matter of a straightforward extension of any given rule, or the addition to the grammar of one more rule recognizing another pattern. For example, with regard to extent recognition, event-based expressions are particularly problematic. In Chapter 5 we showed that pattern-based systems perform very badly on such examples. Approaches using syntactic information, both dependency-based and constituency-based, are significantly better. However, these approaches do less well on the more pattern-based expressions. The question is then how to combine such an approach with a pattern-based system, i.e. how to decide which expression should be recognized with which approach.

Difficult cases for interpretation include:

- Non-specific expressions, as in the following example:

  (7.1) i'm amazed at how unaffected i am by things, how i'm still coming to work and doing the same things and going for a walk in *the morning*.

  Here, the expression *the morning* is used generically and does not refer to any specific date.

- Finding the anchor of a duration is a non-trivial task in the general case; theoretically, any point expression of proper granularity appearing in anywhere in the text (i.e. either before or after the period expression) may serve as an anchor. It is also not possible to tell just by analysing the lexical content of a period expression whether the annotation requires the anchor to be found, or whether it should remain an unanchored period.

- Resolving complex cases of temporal focus tracking: our current approaches of using the document time-stamp or the most recent point expression work very well, but they are not based on any analysis of text meaning or discourse structure and they do result in errors in some situations.

Addressing all these issues requires more advanced text analysis and understanding incorporating discourse analysis, event recognition, and reasoning.

7.2.2.3 Errors in Gold-Standard Annotations

While carrying out the error analysis for the results obtained for the ACE 2005 Training corpus, we noticed that, in a number of cases, the source of disagreement between the system-generated annotations and those found in the gold standard was not due to errors made by the system, but due to errors in the gold-standard annotations. The corpus is, of course, of considerable size (599 documents, 260k words, and 5469 TIMEX2 annotations), and human annotation is a time-consuming and error-prone process. Annotation errors may arise for a number of reasons; Ferro (2004) discusses these in the context of the preparation of the ACE 2004 TERN data. We summarize here the major issues we observed for the ACE 2005 Training corpus.

False Positive Annotations We found a number of words that are annotated as temporal expressions, although they should not be considered as such according to the TIMEX2 guidelines. This includes, for example, the words *new*, *old* and *already.* The annotations of *new* alone caused DANTE to 'miss' 56 annotations.

Similarly, in the document UN/ALT.SUPPORT.DIVORCE there are 20 annotations of *ex* as temporal expressions, which results in 20 missing annotations from a single document. This includes both *ex* being part of words like *exwife* and *exspouse* but also *ex* used on its own as a noun (as in *My ex called and told me...*).

Incorrect Extent The TIMEX2 guidelines are very clear about not including temporal prepositions in the extent of markable expressions (see, e.g. (Ferro et al., 2005, p. 57)). This rule, however, is often not obeyed in the gold-standard annotation of the corpus.

Other extent errors arise as a result of words being omitted in the annotations: for example, *old* missing in *sixteen year old* or *good* in *Good Morning* (the TIMEX2 guidelines actually provide examples of these expressions; see (Ferro et al., 2005, pp. 47, 58)). Similarly, many instances of a half-day designator (e.g. *a.m.*) are missing the final character, although the examples in the TIMEX2 guidelines suggest that the whole designator should be included in the extent (see (Ferro et al., 2005, pp. 16, 63)).

In another case, two expressions were incorrectly annotated with a single annotation (additionally, including the leading preposition):

(7.2) Adnan Pachachi, a onetime foreign minister who returned to Iraq *on May 6 after 33 years* in exile ...

Incorrect Semantic Value Example (7.3) presents a fragment of a document with the gold-standard annotations marked in italics. The expression *the thirtieth* refers to the 30th day of a month; however, it was annotated as the 13th. In some circumstances it is conceivable that the annotator may have decided that the expression was misspelt, and might assign a correct value on the basis of other evidence from the context; but here, *Monday* was annotated as `2004-12-20` and *ten days* as `VAL=P10D ANCHOR_DIR=STARTING ANCHOR_VAL=2004-12-20`, so the expression in question should have the value `2004-12-30`, not `2004-12-13`.

(7.3) S1: Yeah. I'm leaving on *Monday* and coming back on *the thirtieth*, so um,
S2: That is great.
S1: yeah, *ten days*. I'm so excited.

There are also examples of using a format which is not a well-formed TIMEX2 value. For example, one instance of the expression *an extra hour* received the value `T1` when it should be `PT1H`; even if it was correct to use the time format instead of the duration format, the value should be `T01`. Other ill-formed values we found were, for example, `PDE` or `2003TNI` (four occurrences in a single document).

Lack of Consistency Even if the annotators of the corpus took a different approach to certain types of temporal expressions than is suggested in the TIMEX2 guidelines, it would be desirable to be consistent, so that a tagging system could be adapted to the new requirements. Unfortunately, we noticed that in the ACE 2005 corpus there are number of examples of inconsistency.

One example is the treatment of time-stamp expressions like *????-??-??T15:39:00*. In some cases the date part filled with question marks is included in the extent, but in others it is not. It is simply impossible to prepare a recognition grammar that would get all these cases correct.

The TIMEX2 guidelines explicitly state that age in the pattern PERSON_NAME, AGE_NUMBER, is not to be annotated, because there is no trigger (Ferro et al., 2005, p. 20). We found that in the corpus there are indeed 22 such occurrences which are not annotated, but there are another 11 cases that did receive an annotation.

Again, there is not a consistent position taken across the corpus on the queston of how to annotate certain keywords. For example, *holiday* is sometimes annotated, but sometimes it is not. This variation happens even on identical sentences that are repeated (e.g. discussion prompts).

Another example is the annotation of *Christmas*. An extreme case we found is the sentence shown in Example (7.4); here, two occurrences of *Christmas Eve* are annotated (here, marked in italics), but one is missed.

(7.4) Mm mm. Well, we — *Christmas Eve* is *the big day* for — but not Christmas Eve because we always got together on *Christmas Eve*.

Repeated Text This is not an annotation error as such, but it is a characteristic of the data that impacts the evaluation results. We noticed that there are several different documents, for example in the telephone conversations domain, which contain the same fragment; this may be, for example, the same prompt for a given topic of a discussion. These prompts are usually a few sentences in length, and therefore may contain a number of temporal expressions. Success or failure in recognizing these particular expressions is thus magnified.

7.3 Conclusions

With the experience gained when carrying out the work presented in the preceding chapters of this thesis, we have developed DANTE, a new system for processing temporal expressions. This chapter focused on the technical details involved in the development of the tagger and discussed its performance based on the evaluation on three benchmark corpora.

DANTE is a rule-based system, implemented in a pipeline architecture as a plugin to the GATE platform. Taking a knowledge-engineering approach to system development provides us with high level of control over the system's behaviour and does not make the system dependent on large volumes of training data. By using the GATE platform we obtain ready-made solutions for document and annotation representations. It also facilitates integration with other processing components: both the preprocessing components used in the DANTE's pipeline, and those that might want to use DANTE in their pipelines.

Two main modules in DANTE are responsible for the processing of temporal expressions: a recogniser and an interpreter. The former detects expressions in texts, determines their textual extents, and extracts their local semantics, representing this information in our LTIMEX scheme. The latter reads the output of the recogniser and, using the context of the whole document, outputs the global semantics of the expressions represented in TIMEX2.

Based on evaluations carried out on popular benchmarks used in the community, the system delivers state-of-the-art performance and achieves better results than those published in the literature: the ACE Values for the ACE 2005 Training, ACE 2007 Evaluation and WikiWars corpora are 61.9, 67.6 and 81.0, respectively. Further development of DANTE is possible and planned, as there is still room for improvement, both in the area of recognition and interpretation.

Chapter 8

Conclusions

In this final chapter we conclude the work presented in this thesis. First, in Section 8.1, we summarise our work; we review the tasks and experiments we have designed and carried out, collate the results obtained and list the resources that have been developed in the course of this work. We also indicate how the outcomes of our research correspond to the objectives that we set at the outset.

Then, in Section 8.2, we sketch the directions that could be taken for future research that could improve or extend the work presented in this thesis.

MOSCOW, `<TIMEX2 val="2003-04-04">`2003-04-04`</TIMEX2>`

Russia has accepted a US$150 million World Bank loan to combat the spread of AIDS and tuberculosis, ending a negotiating process that lasted `<TIMEX2 val="P4Y" anchor_dir="ENDING" anchor_val="2003">`four years`</TIMEX2>`, World Bank officials said `<TIMEX2 val="2003-04-04">`Friday`</TIMEX2>`. We expected such a decision based on some unofficial comments made `<TIMEX2 val="2003-04-03TEV">`the previous evening`</TIMEX2>` by some of those involved in the negotiations.

Figure 8.1: An example text with inline TIMEX2 annotations.

8.1 Summary of the Contributions and Outcomes

The work presented in this thesis concerns the recognition and interpretation of temporal expressions. For our purposes, we defined a temporal expression as a linguistic expression referring to a temporal entity, of which we distinguished two types: points in time and periods; we also considered temporal expressions referring to sets of such entities. Here, a point in time is not a durationless abstract point, but a point on a timeline which represents a chunk of time that has a duration of one temporal unit, such as a minute, a day or a year; the length of the unit determines the granularity of the points on the timeline. A period is an entity which has a duration of some number of temporal units (for example, five days), possibly mixing different temporal units (for example, two minutes and thirteen seconds) and which is used to describe the length of processes, rather than their temporal location on a timeline, as is the case with point entities.

More specifically, our work is situated in the field of research known as information extraction, and for the purposes of evaluation we adopt the task of extracting temporal expressions as was defined for the TERN task at the ACE evaluations. This requires each temporal expression in a text to be found and annotated with an inline TIMEX2 tag. This tag is used to mark up the textual extent of the expression and to represent its meaning by a set of semantics-related attributes as specified in the TIMEX2 guidelines of 2005. Some examples are shown in Figure 8.1.

Although, ultimately, our goal was to develop a broad-coverage temporal expression tagger capable of finding in texts temporal expressions and interpreting them, our work was not purely an engineering exercise focused on the implementation of a software system. While the performance of the resulting system is a key focus, we were also interested in researching what forms temporal expressions take in real texts, what issues arise in their processing and whether there are any new approaches to the specific problems involved that can facilitate the development of the tagger.

We will now summarize the outcomes and achievements of the work.

8.1.1 The WikiWars Corpus

One of the outcomes that arises from the work carried out to prepare the thesis is a new corpus containing annotated temporal expressions, which we called WikiWars; this was introduced in Chapter 3.

The corpus contains 2681 annotations of temporal expressions, which makes it comparable in size to other corpora. The documents found in WikiWars are historical narratives, sourced from Wikipedia, which describe the temporal progression of military conflicts. The narrative genre has not been included before in any other corpus used in research on temporal information extraction, and the documents exhibit a more interesting discourse structure than is typically found in the currently available corpora.

The corpus has been released to the public, and has already been used by the members of the research community, for example by Alonso et al. (2011), Strötgen and Gertz (2011a) and Strötgen and Gertz (2011b).

8.1.2 The Taxonomy

To grasp the variety of temporal expressions, we started by investigating the range of temporal expressions we find in real texts and how these types are related to each other. We compared our observations to various taxonomies found in the literature, and this resulted in the creation of a comprehensive taxonomy of temporal expressions, which we presented in Chapter 4.

We drew top-level distinctions between temporal expressions referring to single and multiple temporal entities, and between point and period forms of these entities. We then further divided these types into a number of subtypes.

The presented taxonomy provides a more detailed analysis of the range of temporal expressions than any to be found in the existing literature. It also addresses some problems with existing accounts, and in particular proposes a unified use of terminology, which currently largely differs between various authors. The distinctions we presented are useful for identification of the problems involved in the interpretation of temporal expressions; it also has impact on the subsequent design of our representation of local semantics (LTIMEX) and the processing flow in the DANTE tagging system described later in the thesis.

8.1.3 Temporal Expression Recognition

Once a temporal expression is detected by finding an occurrence of a trigger, we must decide which of the surrounding tokens should also be considered part of the temporal expression. Since temporal expressions are syntactic constituents of sentences, a promising approach to extent recognition is to employ syntactic information: because the development of a pattern-based grammar for the broad-coverage recognition of temporal expressions is a laborious task, the expected benefit of any syntax-based approach is a reduction in the development effort. In Chapter 5 we presented our experimentation with two new approaches developed for this purpose: a dependency-based approach and a constituency-based approach. Altogether, we experimented with seven different well-known and state-of-the-art parsers: Minipar, Connexor, Stanford, C&C, Bikel/ Collins, Charniak, and Charniak-Johnson. Overall, the constituency-based approach performed better in our experiments than the dependency-based approach, although for some set-ups the difference in F-measure was very small or there was no difference at all; based on the obtained results, it would be a little premature to decide at this point which of the two types of syntactic analysis is generally more useful for the recognition of temporal expressions.

Although the results obtained with both approaches are reasonably high, they are lower than those obtained with our pattern-based DANTE tagger. However, if we compare the results only for event-based expressions, it turns out that both the dependency-based and constituency-based approaches were vastly superior to the pattern-matching approach.

The most significant source of errors is the quality of syntactic parsing; a common concern where parsers provided an incorrect analysis of a sentence is the attachment of prepositional phrases, which is particularly problematic because many temporal expressions involve these constructions.

8.1.4 The Representation of Local Semantics

In the area of the meaning representation for temporal expressions we proposed in Chapter 6 a representation for the context-independent semantics of temporal expressions. Since this level of representation does not involve any use of context, we call this semantics 'local', and we refer to this representation as LTIMEX. As LTIMEX is essentially compatible with the TIMEX2 and TimeML standards—both of which assume annotations of temporal expressions with the representation of the global (i.e. interpreted in the context of the use) semantics only—our representation is also string-based and the values are in a format derived from the ISO 8601 standard for representation of temporal information. The design of LTIMEX makes this representation intuitive for a human annotator and also makes it easy for a human to compare the local and global values. Another advantage of LTIMEX is the possibility of its application to a wide range of types of temporal expressions.

LTIMEX provides a means of support for an additional level of semantic representation; this in turn supports a modular design for temporal expression tagging, with a well-defined interface between the recognition and interpretation modules, and consequently allows for a more detailed evaluation of taggers.

8.1.5 Temporal Focus Tracking

Temporal focus tracking concerns finding in the text a temporal expression whose value is to be used as the reference time to interpret a context-dependent temporal expression. Although the descriptions of temporal expression taggers found in the literature sometimes mention how the reference time is determined, most often there is no direct evaluation of the chosen solution and in consequence no conclusions can be drawn about the particular methods adopted. One of the reasons for such a situation is that none of the available annotated corpora facilitates the evaluation of temporal focus tracking algorithms. Therefore we prepared the required data ourselves and ran comparative evaluations of various heuristics, the results of which are presented in Chapter 6.

We found that two basic approaches mentioned in the literature—using the document time-stamp (DCD) and using the most recent point temporal expression (MRP)—can work very well for some datasets, but perform not so well or even do not work at all for other datasets. However, for any given dataset used in our experiments the performance of the two methods differed significantly.

A more sophisticated approach is to first identify deictic expressions (i.e. those which are dependent on the utterance time) and to use the time-stamp only for the interpretation of these, but to use the most recent expression for all the other context-dependent expressions. We determined that the classification rules used in the state-of-the-art Chronos tagger worked very well for most of the ACE domains, but performed poorly for the WikiWars corpus. The underlaying problem was the assumption that bare weekday and month names are always deictic, but this is clearly not the case. Therefore we improved this classification and for a dataset combining all the documents used in the experiments we obtained the best result of all of the tested approaches.

8.1.6 Interpretation of Bare Weekday Names

Another problem that we investigated in Chapter 6 was the interpretation of bare weekday names. These expressions, similarly as other underspecified temporal expressions, constitute a special case, in that they do not specify the direction of interpretation from the reference time. For example, *Tuesday* could refer to a day that is in the past or in the future with respect to the reference time, or it could refer to the reference time, but at a different granularity level.

We first experimented with a number of window-based heuristics; here, a calendar window of seven consecutive days is used and we choose the date from this window that is denoted by the weekday name. The differences appear from using different windows in relation to the reference day; the best performing solution turned out to be the window where the reference time is the middle day.

A more sophisticated approach is to use the tense of what we refer to as the controlling verb. We tried different strategies that attempt to determine which verb in the sentence governs the temporal expression, and use its tense to decide in which direction the offset should be made. At first the results turned out to be lower than with using the best-performing window-based method. However, we observed that there were cases when the tense-based methods provided correct results but the window-based method did not work. Therefore we combined the two approaches into a hybrid algorithm: we first find the tense-based evidence within a very local context, and if no verbs are found, we then apply the window method. With such a combination we achieved the best performance of all the tested methods.

8.1.7 The Tagging System

We also developed a complete tagging system, called DANTE, which we present in detail in Chapter 7. The system is rule-based and has a modular architecture with a clear separation into two main processing components as follows: (1) recognition and context-independent semantic processing, and (2) interpretation in the context of the content of the whole document.

The recognition module reads in preprocessed documents; these already contain annotations of sentences, tokens with POS information, named entities and gazetteer lookups. The module is implemented as a cascaded finite-state transducer consisting of about 350 JAPE rules and 100 macros. These rules recognize chunks of texts that constitute temporal expressions and generate TIMEX2 annotations which mark up their extents in text and encode their local semantics in our LTIMEX notation. The performance of the recognizer is state-of-the-art; the F-measure for the detection task currently is 0.91 for the ACE 2005 Training corpus and 0.98 for the WikiWars corpus; for the recognition task the corresponding F-measures are 0.79 and 0.93.

The interpretation module iterates through all of the recognized expressions, reads in their local semantics representations, and carries out further processing. In particular, depending on the type of the expression, different operations are applied to derive the representation of global meaning. If necessary, the module uses calendar arithmetics, unification of semantic values, a temporal focus tracking mechanism, and an algorithm for determining the direction of interpretation of bare weekday names. The global semantics value is then stored in the TIMEX2 attributes of the annotation.

Currently, the overall performance of the system measured with the ACE value is 67.6 for the ACE 2007 Evaluation corpus and 81.0 for WikiWars, which exceeds the results reported for other systems in the literature.

DANTE, which we are making publicly available,[1] is implemented as a plugin to the GATE platform, which allows for particularly easy integration in any application based on GATE.

8.1.8 A Review of the Aims

In Chapter 1 we identified three areas of research that we planned to address in our work; these were:

1. The taxonomization of temporal expressions, with the purpose of classifying and describing the full range of temporal expressions that are found in texts, and the development of a representation of their semantics which is appropriate for implementation in software.

2. The development of annotation schemes, with the aim of providing annotations capable of expressing the required semantics, and the construction of annotated corpora, to provide development and evaluation datasets.

3. The design and implementation of algorithms for temporal expression tagging which are robust, efficient, perform well, and provide wide coverage of the temporal expressions found in text, including their semantic annotation.

We have contributed to these areas in the following way:

1. We developed a rich taxonomy of temporal expressions that underlies the work in the thesis. We also proposed a string-based representation for local semantics.

2. We demonstrated that our representation for local semantics can be integrated with the existing annotation schemes, thus contributing to further development in this area. We also constructed a new annotated corpus, which substantially differs from existing corpora in terms of the genre of text it contains.

3. We explored new syntax-based methods for extent recognition, showing that there is a class of temporal expressions the recognition of which these methods are particularly well suited to. We also experimented with various heuristics for determining the direction of interpretation of bare weekday names and for selecting a reference time from the remaining document context for the interpretation of context-dependent expressions. Finally, we developed a fully-featured temporal expression tagger whose coverage and tagging performance exceeds the results reported in the literature.

Our contributions thus constitute six major outcomes of the thesis: (1) the taxonomy, (2) the LTIMEX annotation scheme, (3) syntax-based extent recognition methods, (4) advances in the interpretation of temporal expressions, (5) a new complete tagging system, and (6) a new annotated corpus.

[1]The system can be downloaded from `http://timexportal.wikidot.com/dante`.

8.2 Directions for Future Research

The work presented in this thesis can be continued by undertaking new research that can either further contribute to the extraction of temporal expressions, or extend to a further level of document processing; we outline some possible directions below.

8.2.1 Addressing Requirements of Specific Domains

We note that processing temporal information in selected domains may require special approaches. One such domain is that of patient medical records. Zhou and Hripcsak (2007) reviewed the state-of-the-art in temporal reasoning for this domain and suggested that the temporal expressions found here are quite specific; for example, *post-op # 6* means 'the sixth day after an operation' and *q.i.d.* means 'four times a day'. Also, such texts often do not consist of proper English sentences, which is an additional challenge to the use of existing text processing techniques.

8.2.2 Set Expressions

Set expressions are underdeveloped in our work, just as they are in the work of others that takes the TIMEX2 standard as its starting point. First of all, the definition of set expressions should be revisited; for example, *Monday and Tuesday* is treated in TIMEX2 as two point expressions, but we argue there are reasonable grounds on which such strings could be considered as single expressions referring to a set of entities. Secondly, without a precise definition of what set expressions are, taxonomising them is in turn also quite obscured. Advances in these two areas would provide grounds for the further development of semantic representations and annotation schemes for set expressions.

8.2.3 Geotagging and Normalization to a Common Time Zone

The interpretation step could be augmented with the final normalisation of all mentioned times into a single time zone. Doing this at an adequate level requires detecting the geographic area which the document assumes. However, unless rich enough metadata are provided with the processed document, we note that such capabilities go beyond the standard task of processing temporal expressions, and require a sophisticated analysis of the document content and its meaning. Extensions to support daylight savings time should also be added.

8.2.4 Addition of Event Recognition

Extending the processing of temporal expressions with event recognition and event time-stamping would increase the usability of the tool for many applications requiring natural language processing, e.g. question answering or document summarisation. Since TimeML facilitates such a combination for inline annotations, adaptation of DANTE to produce TimeML annotations may be the first step towards extending the capabilities of the system.

8.2.5 Going Multilingual

Multilingual temporal tagging, or even more broadly, multilingual information extraction, is a direction currently attracting a lot of attention in the research community. Adjusting DANTE to a new language, or a number of languages, would be a very valuable step forward in its development. This may, however, require first to prepare an adjusted version of annotation guidelines and new training and evaluation corpora.

```
<TempEx val="$X">The End.</TempEx>
```

References

D. Ahn, S. F. Adafre, and M. de Rijke. 2005a. Extracting temporal information from open domain text: A comparative exploration. In R. van Zwol, editor, *Proceedings of the 5th Dutch-Belgian Information Retrieval Workshop (DIR)*, pages 3–10, Utrecht, The Netherlands, January. Center for Content and Knowledge Engineering. Republished in: Journal of Digital Information Management, 3(1):14–20, 2005.

D. Ahn, S. F. Adafre, and M. de Rijke. 2005b. Recognizing and interpreting temporal expressions in open domain texts. In S. N. Artëmov, H. Barringer, A. S. d'Avila Garcez, L. C. Lamb, and J. Woods, editors, *We Will Show Them: Essays in Honour of Dov Gabbay, Vol 1*, pages 31–50, October.

D. Ahn, J. van Rantwijk, and M. de Rijke. 2007. A cascaded machine learning approach to interpreting temporal expressions. In *Proceedings of Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT)*, pages 420–427, Rochester, NY, USA, April. Association for Computational Linguistics.

A. Akmajian, R. A. Demers, A. K. Farmer, and R. M. Harnish. 1990. *Linguistics: An Introduction to Language and Communication.* MIT Press, third edition.

J. Alexandersson, N. Reithinger, and E. Maier. 1997. Insights into the dialogue processing of VERBMOBIL. In *Proceedings of the Fifth Conference on Applied Natural Language Processing (ANLP)*, pages 33–40, Washington, DC, USA, March. Association for Computational Linguistics.

J. Alexandersson, R. Engel, M. Kipp, S. Koch, U. Küssner, N. Reithinger, and M. Stede. 2000. Modeling negotiation dialogs. In W. Wahlster, editor, *Verbmobil: Foundations of Speech-to-Speech Translation*, pages 441–451. Springer.

J. F. Allen. 1981. An interval-based representation of temporal knowledge. In P. J. Hayes, editor, *Proceedings of the 7th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 221–226, Vancouver, BC, Canada, August. William Kaufmann.

J. F. Allen. 1983. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843, November.

J. F. Allen. 1984. Towards a general theory of action and time. *Artificial Intelligence*, 23(2):123–154.

J. F. Allen. 1995. *Natural Language Understanding.* The Benjamin/Cummings Publishing Company, Inc.

O. Alonso, J. Strötgen, R. Baeza-Yates, and M. Gertz. 2011. Temporal information retrieval: Challenges and opportunities. In R. Baeza-Yates, J. Masanès, and M. Spaniol, editors, *Proceedings of the 1st International Temporal Web Analytics Workshop (TWAW)*, pages 1–8, Hyderabad, India, March.

O. R. Alonso. 2008. *Temporal Information Retrieval.* Ph.D. thesis, University of California, August.

D. E. Appelt, J. R. Hobbs, J. Bear, D. Israel, M. Kameyama, A. Kehler, D. Martin, K. Myers, and M. Tyson. 1995. SRI International FASTUS System. MUC-6 test results and analysis. In *Proceedings of the Sixth Message Understanding Conference (MUC-6)*, pages 237–248, Columbia, Maryland, November. Morgan Kaufmann.

E. Bach. 1986. The algebra of events. *Linguistics and Philosophy*, 9:5–16.

E. Bach. 2005. The algebra of events. In I. Mani, J. Pustejovsky, and R. Gaizauskas, editors, *The Language of Time: A Reader*, chapter 3, pages 61–70. Oxford University Press.

J. Baldwin. 2002. Learning temporal annotation of French news. Master's thesis, Dept. of Linguistics, Georgetown University, April.

A. Bassara. 2008. *Temporalne indeksowanie dokumentów dla analizy rynków finansowych.* Ph.D. thesis, Akademia Ekonomiczna w Poznaniu, June.

D. Battistelli, M. Cori, J.-L. Minel, and C. Teissèdre. 2011. Semantics of calendar adverbials for information retrieval. In M. Kryszkiewicz, H. Rybinski, A. Skowron, and Z. Ras, editors, *Foundations of Intelligent Systems*, volume 6804 of *LNAI*, pages 622–631. Springer-Verlag.

M. R. Bennett and B. H. Partee. 1978. *Toward the Logic of Tense and Aspect.* Indiana University Linguistics Club (Bloomington).

A. Berglund, R. Johansson, and P. Nugues. 2006. A machine learning approach to extract temporal information from texts in Swedish and generate animated 3D scenes. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 76–86, Trento, Italy, April. Association for Computational Linguistics.

A. Berglund. 2004. Extracting temporal information and ordering events for Swedish. Master's thesis, Lund University, October.

S. Bethard and J. H. Martin. 2006. Identification of event mentions and their semantic class. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 146–154, Sydney, Australia, July. Association for Computational Linguistics.

C. Bettini, S. Jajodia, and S. X. Wang. 2000. *Time Granularities in Databases, Data Mining and Temporal Reasoning.* Springer-Verlag.

A. Bies, M. Ferguson, K. Katz, R. MacIntyre, V. Tredinnick, G. Kim, M. A. Marcinkiewicz, and B. Schasberger. 1995. Bracketing guidelines for Treebank II style: Penn Treebank project. Technical report, University of Pennsylvania, January.

D. M. Bikel, R. Schwartz, and R. M. Weischedel. 1999. An algorithm that learns what's in a name. *Machine Learning*, 34:211–231.

D. M. Bikel. 2002. Design of a multi-lingual, parallel-processing statistical parsing engine. In *Proceedings of the Second International Conference on Human Language Technology Research (HLT)*, pages 178–182, San Diego, CA, USA, March. Morgan Kaufmann Publishers Inc.

A. Bittar. 2009. Annotation of events and temporal expressions in French texts. In *Proceedings of the ACL 2009 Third Linguistic Annotation Workshop (LAW)*, pages 48–51, Suntec, Singapore, August. Association for Computational Linguistics.

B. Boguraev, J. Castaño, R. Gaizauskas, B. Ingria, G. Katz, B. Knippen, J. Littman, I. Mani, J. Pustejovsky, A. Sanfilippo, A. See, A. Setzer, R. Saurí, A. Stubbs, B. Sundheim, S. Symonenko, and M. Verhagen. 2005. TimeML 1.2.1 – a formal specification language for events and temporal expressions. http://timeml.org/site/publications/timeMLdocs/timeml_1.2.1.html, October.

B. Boguraev, J. Pustejovsky, R. Ando, and M. Verhagen. 2007. TimeBank evolution as a community resource for TimeML parsing. *Language Resources and Evaluation*, 41(1):91–115, February.

S. Busemann, S. Oepen, E. Hinkelman, G. Neumann, and H. Uszkoreit. 1994. COSMA – multi-participant NL interaction for appointment scheduling. Research Report RR-94-34, DFKI, Saarbrücken, October.

S. Busemann, T. Declerck, A. K. Diagne, L. Dini, J. Klein, and S. Schmeier. 1997. Natural language dialogue service for appointment scheduling agents. In *Proceedings of the Fifth Conference on Applied Natural Language Processing (ANLP)*, pages 25–32, Washington, DC, USA, March. Association for Computational Linguistics.

J. Carroll, T. Briscoe, and A. Snfilippo. 1998. Parser evaluation: a survey and a new proposal. In *Proceedings of the 1st International Conference on Language Resources and Evaluation (LREC)*, pages 447–454, Granada, Spain, May. European Language Resources Association.

T. Caselli, F. dell'Orletta, and I. Prodanof. 2009. TETI: a TimeML compliant TimEx tagger for Italian. In *Proceedings of the International Multiconference on Computer Science and Information Technology (IMCSIT)*, pages 185–192, Mragowo, Poland, October.

E. Charniak and M. Johnson. 2005. Coarse-to-fine n-best parsing and MaxEnt discriminative reranking. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 173–180, Ann Arbor, Michigan, USA, June. Association for Computational Linguistics.

E. Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of the 1st North American Chapter of the Association for Computational Linguistics Conference (NAACL)*, pages 132–139, Seattle, Washington, USA, April. Morgan Kaufmann Publishers Inc.

N. Chinchor and P. Robinson. 1998. MUC-7 named entity task definition (version 3.5). In *Proceedings of the 7th Message Understanding Conference (MUC-7)*, Fairfax, Virginia.

S. Clark and J. R. Curran. 2007. Wide-coverage efficient statistical parsing with CCG and log-linear models. *Computational Linguistics*, 33(4):493–552, December.

R. Crouch, R. M. Kaplan, T. H. King, and S. Riezler. 2002. A comparison of evaluation metrics for a broad-coverage stochastic parser. In J. Carroll, editor, *Proceedings of the LREC 2002 Workshop: Beyond Parseval – Towards Improved Evaluation Measures for Parsing Systems*, pages 67–74, Las Palmas, Canary Islands, Spain, May.

H. Cunningham, D. Maynard, and V. Tablan. 2000. JAPE: a Java Annotation Patterns Engine (second edition). Technical Report CS–00–10, Department of Computer Science, University of Sheffield.

H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan. 2002. GATE: A framework and graphical development environment for robust NLP tools and applications. In *Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics (ACL)*, pages 168–175, Philadelphia, Pennsylvania, USA, July. Association for Computational Linguistics.

I. Dagan, O. Glickman, and B. Magnini. 2006. The PASCAL recognising textual entailment challenge. In J. Quiñonero-Candela, I. Dagan, B. Magnini, and F. d'Alché Buc, editors, *Machine Learning Challenges. Evaluating Predictive Uncertainty, Visual Object Classification, and Recognizing Textual Entailment (First PASCAL Machine Learning Challenges Workshop, MLCW, Southampton, UK, April 2005, Revised Papers)*, volume 3944 of *LNAI*, pages 177–190. Springer-Verlag.

D. Day, C. McHenry, R. Kozierok, and L. Riek. 2004. Callisto: A configurable annotation workbench. In *Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC)*, pages 2073–2076, Lisbon, Portugal, May. European Language Resources Association.

M.-C. de Marneffe and C. D. Manning. 2008. Stanford typed dependencies manual. `http://nlp.stanford.edu/software/dependencies_manual.pdf`, September.

M.-C. de Marneffe, B. MacCartney, and C. D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC)*, pages 449–454, Genoa, Italy, May. European Language Resources Association.

L. Derczynski and R. Gaizauskas. 2010. USFD2: Annotating temporal expresions and TLINKs for TempEval-2. In *Proceedings of the 5th International Workshop on Semantic Evaluation (SemEval)*, pages 337–340, Uppsala, Sweden, July. Association for Computational Linguistics.

L. Derczynski and R. Gaizauskas. 2011. An annotation scheme for Reichenbach's verbal tense structure. In *Proceedings of the 6th Joint ACL – ISO Workshop on Interoperable Semantic Annotation*, pages 10–17, Oxford, UK, January.

D. Dowty. 1979. *Word Meaning and Montague Grammar*. Springer.

A. Elkhlifi and R. Faiz. 2010. Event extraction approach for Web 2.0. In *Proceedings of the ACS/IEEE International Conference on Computer Systems and Applications (AICCSA)*, pages 1–8, Hammamet, Tunisia, May. Institute of Electrical and Electronics Engineers.

U. Endriss. 1998a. Semantik Zeitlicher Ausdrücke in Terminvereinbarungsdialogen (semantics of temporal expressions in appointment scheduling dialogues). Verbmobil Report 227, Technical University Berlin, Department of Computer Science, August.

U. Endriss. 1998b. Zeitliche Ausdrücke in Terminvereinbarungsdialogen: Repräsentation und Inferenz (Temporal expressions in appointment scheduling dialogues: Representation and inference). Diploma thesis, Technical University Berlin, Department of Computer Science, June.

J. Euzenat. 1995. An algebraic approach to granularity in qualitative time and space representation. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 894–900, Montreal, Canada, August. Morgan Kaufmann Publishers Inc.

V. Evans. 2004. *The Structure of Time.* Human Cognitive Processing 12. John Benjamins, Philadelphia.

L. Ferro, I. Mani, B. Sundheim, and G. Wilson. 2001. TIDES temporal annotation guidelines – version 1.0.2. Technical Report MTR 01W0000041, MITRE, June.

L. Ferro, L. Gerber, I. Mani, B. Sundheim, and G. Wilson. 2004. TIDES 2003 standard for the annotation of temporal expressions. Technical report, MITRE, April. http://fofoca.mitre.org/annotation_guidelines/2003_timex2_standard_v1_3.pdf.

L. Ferro, L. Gerber, I. Mani, B. Sundheim, and G. Wilson. 2005. TIDES 2005 standard for the annotation of temporal expressions. Technical report, MITRE, September. http://fofoca.mitre.org/annotation_guidelines/2005_timex2_standard_v1.1.pdf.

L. Ferro. 2001. Instruction manual for the annotation of temporal expressions. Technical Report MTR 01W0000046V01, MITRE, June.

L. Ferro. 2004. Annotating the TERN corpus. In *Proceedings of Temporal Expression Recognition and Normalization (TERN) Evaluation Workshop*, September. http://timex2.mitre.org/tern_2004/ferro2_TERN2004_annotation_sanitized.pdf.

E. Filatova and E. Hovy. 2001. Assigning time-stamps to event-clauses. In L. Harper, I. Mani, and B. Sundheim, editors, *Proceedings of the ACL 2001 Workshop on Temporal and Spatial Information Processing*, pages 1–8, Toulouse, France, July. Association for Computational Linguistics.

M. Fisher, D. Gabbay, and L. Vila, editors. 2005. *Handbook of Temporal Reasoning in Artificial Intelligence.* Elsevier Science Inc.

A. Galton. 1995. Time and change for AI. In D. M. Gabbay, C. J. Hogger, and J. A. Robinson, editors, *Handbook of Logic in Artificial Intelligence and Logic Programming*, volume IV, pages 175–240. Oxford University Press.

B. J. Grosz and C. L. Sidner. 1986. Attention, intentions, and the structure of discourse. *Computational Linguistics*, 12(3):175–204.

B. J. Grosz. 1977. The representation and use of focus in a system for understanding dialogs. In R. Reddy, editor, *Proceedings of the 5th International Joint Conference on Artificial intelligence (IJCAI)*, volume 1, pages 67–76, Cambridge, MA, USA, August. Morgan Kaufmann Publishers Inc.

K. Hacioglu, Y. Chen, and B. Douglas. 2005. Automatic time expression labeling for English and Chinese text. In A. F. Gelbukh, editor, *Proceedings of the 6th International Conference on Intelligent Text Processing and Computational Linguistics (CICLing)*, volume 3406 of *LNCS*, pages 548–559, Mexico City, Mexico, February. Springer-Verlag.

E. Hajnicz. 1996. *Time Structures – Formal Description and Algorithmic Representation*, volume 1047 of *LNAI*. Springer-Verlag.

B. Han and M. Kohlhase. 2003. A time calculus for natural language. In P. Blackburn and J. Bos, editors, *Proceedings of the 4th International Workshop on Inference in Computational Semantics (ICoS)*, pages 113–127, Nancy, France, September.

B. Han and A. Lavie. 2004. A framework for resolution of time in natural language. *ACM Transactions on Asian Language Information Processing (TALIP)*, 3(1):11–32, March.

B. Han, D. Gates, and L. Levin. 2005. Anchoring temporal expressions in scheduling-related emails. In G. Katz, J. Pustejovsky, and F. Schilder, editors, *Annotating, Extracting and Reasoning about Time and Events*, number 05151 in Dagstuhl Seminar Proceedings, Wadern-Dagstuhl, Germany, April. Leibniz-Zentrum für Informatik GmbH.

B. Han, D. Gates, and L. Levin. 2006a. Understanding temporal expressions in emails. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics (HLT-NAACL)*, pages 136–143, New York, NY, USA, June. Association for Computational Linguistics.

B. Han, D. Gates, and L. Levin. 2006b. From language to time: A temporal expression anchorer. In *Proceedings of the Thirteenth International Symposium on Temporal Representation and Reasoning (TIME)*, pages 196–203, Budapest, Hungary, June. Institute of Electrical and Electronics Engineers.

E. Hinrichs. 1986. Temporal anaphora in discourses of English. *Linguistics and Philosophy*, 9(1):63–82.

L. Hirschman. 1980. Retrieving time information from natural-language texts. In R. N. Oddy, S. E. Robertson, C. J. van Rijsbergen, and P. W. Williams, editors, *Proceedings of the 3rd Annual ACM Conference on Research and Development in Information Retrieval (SIGIR)*, pages 154–171, Cambridge, UK, June. Butterworth & Co.

J. Hitzeman. 2005. Text type and the position of a temporal adverbial within the sentence. In G. Katz, J. Pustejovsky, and F. Schilder, editors, *Annotating, Extracting and Reasoning about Time and Events*, number 05151 in Dagstuhl Seminar Proceedings, Wadern-Dagstuhl, Germany, April. Leibniz-Zentrum für Informatik GmbH.

J. Hitzeman. 2007. Text type and the position of a temporal adverbial within the sentence. In F. Schilder, G. Katz, and J. Pustejovsky, editors, *Annotating, Extracting and Reasoning about Time and Events (International Seminar, Dagstuhl Castle,* *Germany, April 10-15, 2005, Revised Papers)*, volume 4795 of *LNAI*, pages 29–40. Springer-Verlag.

J. R. Hobbs and F. Pan. 2004. An ontology of time for the semantic web. *ACM Transactions on Asian Language Information Processing (TALIP)*, 3(1):66–85.

J. R. Hobbs and F. Pan. 2006. Time ontology in OWL. Technical report, World Wide Web Consortium (W3C), September. http://www.w3.org/TR/owl-time.

J. R. Hobbs. 1978. Resolving pronoun references. *Lingua*, 44:311–338.

J. R. Hobbs. 1985. Granularity. In *Proceedings of the 9th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 432–435, Los Angeles, CA, USA, August. Morgan Kaufmann Publishers Inc.

G. Hripcsak, N. Elhadad, Y.-H. Chen, L. Zhou, and F. Morrison. 2009. Using empiric semantic correlation to interpret temporal asserions in clinical texts. *Journal of the American Medical Informatics Association*, 16(2):220–227, March.

C. H. Hwang and L. K. Schubert. 1993. Interpreting temporal adverbials. In *Proceedings of the Workshop on Human Language Technology (HLT)*, pages 138–143, Plainsboro, New Jersey, USA, March. Morgan Kaufmann Publishers Inc.

C. H. Hwang and L. K. Schubert. 1994. Interpreting tense, aspect and time adverbials: A compositional, unified approach. In D. M. Gabbay and H. J. Ohlbach, editors, *Proceedings of the First International Conference on Temporal Logic (ICTL)*, volume 827 of *LNAI*, pages 238–264, Bonn, Germany, July. Springer-Verlag.

ISO. 2004. Data elements and interchange formats – information interchange – representation of dates and times. http://www.iso.org/iso/date_and_time_format.

S. Janarthanam, H. Hastie, O. Lemon, and X. Liu. 2011. "The day after the day after tomorrow?" A machine learning approach to adaptive temporal expression generation: training and evaluation with real users. In *Proceedings of the 12th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 142–151, Portland, Oregon, USA, June. Association for Computational Linguistics.

S. B. Jang, J. Baldwin, and I. Mani. 2004. Automatic TIMEX2 tagging of Korean news. *ACM Transactions on Asian Language Information Processing (TALIP)*, 3(1):51–65, March.

H. Kamp. 1979. Events, instants and temporal reference. In R. Bauerle, U. Egli, and A. von Stechow, editors, *Semantics from different points of view*, pages 376–417. Springer-Verlag.

L. Karttunen. 2001. Applications of finite-state transducers in natural language processing. In S. Yu and A. Paun, editors, *Implementation and Application of Automata*, volume 2088 of *LNCS*, pages 34–46. Springer-Verlag.

A. Kenny. 1963. *Action, Emotion and Will.* Routledge and Kegan Paul Limited.

R. Kimura, S. Oyama, H. Toda, and K. Tanaka. 2007. Creating personal histories from the web using namesake disambiguation and event extraction. In L. Baresi, P. Fraternali, and G.-J. Houben, editors, *Proceedings of the 7th International Conference on Web Engineering (ICWE)*, volume 4607 of *LNCS*, pages 400–414, Como, Italy, July. Springer-Verlag.

D. Klein and C. D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting of Association for Computational Linguistics (ACL)*, pages 423–430, Sapporo, Japan, July. Association for Computational Linguistics.

A. Kumar, M. Lease, and J. Baldridge. 2011. Supervised language modeling for temporal resolution of texts. In *Proceedings of the 20th Conference on Information and Knowledge Management (CIKM)*, Glasgow, Scotland, UK, October.

A. Lavelli, B. Magnini, M. Negri, E. Pianta, M. Speranza, and R. Sprugnoli. 2005. Italian Content Annotation Bank (I-CAB): Temporal expressions (v. 1.0). Technical report, ITC-irst, May.

D. D. Lewis, Y. Yan, T. G. Rose, and F. Li. 2004. RCV1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5:361–397, April.

Y. Li, J. Nie, Y. Zhang, B. Wang, B. Yan, and F. Weng. 2010. Contextual recommendation based on text mining. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling)*, volume Posters, pages 692–700, Beijing, China, August. Tsinghua University Pres.

L. B. Lombard. 1978. Actions, results, and the time of a killing. *Philosophia*, 8:341–54.

I. Mani and M. T. Maybury, editors. 1999. *Advances in Automatic Text Summarization*. MIT Press.

I. Mani and G. Wilson. 2000a. Temporal granularity and temporal tagging of text. In *Proceedings of the AAAI 2000 Workshop on Spatial and Temporal Granularity*, pages 71–73, Austin, Texas, July. AAAI Press.

I. Mani and G. Wilson. 2000b. Robust temporal processing of news. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics (ACL)*, pages 69–76, Hong Kong, October. Association for Computational Linguistics.

I. Mani, G. Wilson, L. Ferro, and B. Sundheim. 2001. Guidelines for annotating temporal information. In *Proceedings of the First International Conference on Human Language Technology Research (HLT)*, pages 299–302, San Diego, CA, USA, March. Association for Computational Linguistics.

I. Mani, J. Pustejovsky, and R. Gaizauskas, editors. 2005. *The Language of Time: A Reader*. Oxford University Press, August.

I. Mani. 1998. A theory of granularity and its application to problems of polysemy and underspecification of meaning. In A. G. Cohn, L. K. Schubert, and S. C. Shapiro, editors, *Proceedings of the Sixth International Conference on Principles of Knowledge Representation and Reasoning (KR)*, pages 245–255, Trento, Italy, June.

I. Mani. 2003. Recent developments in temporal information extraction. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP)*, pages 45–60, Borovets, Bulgaria, September.

M. P. Marcus, B. Santorini, and M. A. Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330, June.

C. Masolo, S. Borgo, A. Gangemi, N. Guarino, and A. Oltramari. 2003. WonderWeb deliverable D18: Ontology library (final). Technical Report Del 18, Laboratory For Applied Ontology – ISTC-CNR, December.

P. H. Matthews. 2007. *Concise Dictionary of Linguistics.* Oxford University Press.

M. T. Maybury, editor. 2004. *New Directions in Question Answering.* AAAI Press/MIT Press.

P. Mazur and R. Dale. 2007. The DANTE temporal expression tagger. In Z. Vetulani, editor, *Proceedings of the 3rd Language And Technology Conference (LTC)*, pages 315–319, Poznan, Poland, October. Fundacja Uniwersytetu im. A. Mickiewicza.

P. Mazur and R. Dale. 2008. What's the date? High accuracy interpretation of weekday names. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling)*, pages 553–560, Manchester, UK, August. Coling 2008 Organizing Committee.

D. McDermott. 1982. A temporal logic for reasoning about processes and plans. *Cognitive Science*, 6:101–155.

S. Mizobuchi, T. Sumitomo, M. Fuketa, and J.-I. Aoe. 1998. A method for understanding time expressions. In *Proceedings of IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, volume 2, pages 1151–1155, San Diego, CA, USA, October. Institute of Electrical and Electronics Engineers.

M. Moens and M. Steedman. 1987. Temporal ontology in natural language. In *Proceedings of the 25th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1–7, Stanford, California, July. Association for Computational Linguistics.

M. Moens and M. Steedman. 1988. Temporal ontology and temporal reference. *Computational Linguistics*, 14(2):15–28.

M. Moens and M. Steedman. 2005. Temporal ontology and temporal reference. In I. Mani, J. Pustejovsky, and R. Gaizauskas, editors, *The Language of Time: A Reader*, chapter 6, pages 93–114. Oxford University Press.

M.-F. Moens. 2006. *Information Extraction: Algorithms and Prospects in a Retrieval Context.* The Information Retrieval Series. Springer-Verlag.

A. Mourelatos. 1978. Events, processes and states. *Linguistics and Philosophy*, 2(3):415–434.

A. Nakhimovsky. 1987. Temporal reasoning in natural language understanding: the temporal structure of the narrative. In *Proceedings of the Third Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 262–269, Copenhagen, Denmark, April. Association for Computational Linguistics.

M. Negri and L. Marseglia. 2005. Recognition and normalization of time expressions: ITC-irst at TERN 2004. Technical Report WP3.7, Information Society Technologies, February.

J. Niemi and K. Koskenniemi. 2007. Representing calendar expressions with finite-state transducers that bracket periods of time on a hierarchical timeline. In J. Nivre, H.-J. Kaalep, K. Muischnek, and M. Koit, editors, *Proceedings of the 16th Nordic Conference of Computational Linguistics (NODALIDA)*, pages 355–362, Tartu, Estonia, May. University of Tartu.

J. Niemi and K. Koskenniemi. 2008a. Quantification and implication in semantic calendar expressions represented with finite-state transducers. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling)*, volume Posters and Demonstrations, pages 69–72, Manchester, UK, August. Coling 2008 Organizing Committee.

J. Niemi and K. Koskenniemi. 2008b. Representing and combining calendar information by using finite-state transducers. In J. Piskorski, B. W. Watson, and A. Yli-Jyrä, editors, *Proceedings of the 7th International Workshop on Finite-State Methods and Natural Language Processing (FSMNLP)*, volume 191 of *Frontiers in Artificial Intelligence and Applications*, pages 122–133, Ispra, Italy, September. IOS Press.

NIST. 2007. The ACE 2007 (ACE07) evaluation plan (v1.3a), February. http://www.nist.gov/speech/tests/ace/2007/doc/ace07-evalplan.v1.3a.pdf.

C. Northwood. 2010. TERNIP: Temporal expression recognition and normalisation in Python. Master's thesis, University of Sheffield, September.

H. J. Ohlbach and D. M. Gabbay. 1998. Calendar logic. *Journal of Applied Non-Classical Logics*, 8(4).

H. J. Ohlbach. 2005. Computational treatment of temporal notions – the CTTN-System. In G. Katz, J. Pustejovsky, and F. Schilder, editors, *Annotating, Extracting and Reasoning about Time and Events*, number 05151 in Dagstuhl Seminar Proceedings, Wadern-Dagstuhl, Germany, April. Leibniz-Zentrum für Informatik GmbH.

H. J. Ohlbach. 2007. Computational treatment of temporal notions: The CTTN-System. In F. Schilder, G. Katz, and J. Pustejovsky, editors, *Annotating, Extracting and Reasoning about Time and Events (International Seminar, Dagstuhl Castle, Germany, April 10-15, 2005, Revised Papers)*, volume 4795 of *LNAI*, pages 72–87. Springer-Verlag.

A. K. Orphanides. 2008. Extraction of natural-language dates and comparison of dates in hypothesis and text to identify negative textual entailment. Master's thesis, University of North Carolina, April.

A. Palmer, E. Ponvert, J. Baldridge, and C. Smith. 2007. A sequencing model for situation entity classification. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics (ACL)*, pages 896–903, Prague, Czech Republic, June. Association for Computational Linguistics.

F. Pan and J. R. Hobbs. 2005. Temporal aggregates in OWL-Time. In *Proceedings of the 18th International Florida Artificial Intelligence Research Society Conference (FLAIRS)*, pages 560–565, Clearwater Beach, Florida, May. AAAI Press.

F. Pan, R. Mulkar, and J. R. Hobbs. 2006. Learning event durations from event descriptions. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics (Coling/ACL)*, pages 393–400, Sydney, Australia, July. Association for Computational Linguistics.

F. Pan. 2007. *Representing Complex Temporal Phenomena for the Semantic Web and Natural Language*. Ph.D. thesis, University of Southern California, December.

G. Parent, M. Gagnon, and P. Muller. 2008. Annotation d'expressions temporelles et d'événements en français. In F. Béchet, editor, *Traitement Automatique des Langues Naturelles (TALN)*, Avignon, France, June. Association pour le Traitement Automatique des Langues.

B. Partee. 1973. Some structural analogies between tenses and pronouns in English. *The Journal of Philosophy*, 70(18):601–609, October. Proceedings of the Seventieth Annual Meeting of the American Philosophical Association Eastern Division.

B. Partee. 1984. Nominal and temporal anaphora. *Linguistics and Philosophy*, 7(3):243–286, August.

R. J. Passonneau. 1988. A computational model of the semantics of tense and aspect. *Computational Linguistics*, 14(2):44–60.

R. J. Passonneau. 2005. A computational model of the semantics of tense and aspect. In I. Mani, J. Pustejovsky, and R. Gaizauskas, editors, *The Language of Time: A Reader*, chapter 8, pages 129–158. Oxford University Press.

I. Pratt and N. Francez. 1997. On the semantics of temporal prepositions and preposition phrases. Technical Report LCL9701, University of Manchester, Computer Science Department, June.

J. Preiss. 2003. Using grammatical relations to compare parsers. In *Proceedings of the 10th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, volume 1, pages 291–298, Budapest, Hungary, April. Association for Computational Linguistics.

A. Prior. 1957. *Time and Modality*. Clarendon Press.

G. Puşcaşu. 2004. A framework for temporal resolution. In *Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC)*, pages 1901–104, Lisbon, Portugal, May. European Language Resources Association.

J. Pustejovsky, L. Belanger, J. Castano, R. Gaizauskas, P. Hanks, B. Ingria, G. Katz, D. Radev, A. Rumshisky, A. Sanfilippo, R. Sauri, A. Setzer, B. Sundheim, and M. Verhagen. 2002. TERQAS final report. Technical report, MITRE, September. http://nrrc.mitre.org/NRRC/Docs_Data/TERQAS_2002.

J. Pustejovsky, P. Hanks, R. Saurí, A. See, R. Gaizauskas, A. Setzer, D. Radev, B. Sundheim, D. Day, L. Ferro, and M. Lazo. 2003. The TIMEBANK corpus. In D. Archer, P. Rayson, A. Wilson, and T. McEnery, editors, *Proceedings of the Corpus Linguistics 2003 Conference (CL)*, pages 647–656, Lancaster, UK, March.

J. Pustejovsky, B. Ingria, R. Sauri, J. Castano, J. Littman, R. Gaizauskas, A. Setzer, G. Katz, and I. Mani. 2005. The specification language TimeML. In I. Mani, J. Pustejovsky, and R. Gaizauskas, editors, *The Language of Time: A Reader*, chapter 27, pages 545–558. Oxford University Press.

J. Pustejovsky, K. Lee, H. Bunt, and L. Romary. 2010. ISO-TimeML: An international standard for semantic annotation. In N. Calzolari, K. Choukri, B. Maegaard, J. Mariani, J. Odjik, S. Piperidis, M. Rosner, and D. Tapias, editors, *Proceedings of the 7th International Conference on Language Resources and Evaluation (LREC)*, pages 394–397, Valletta, Malta, May. European Language Resources Association.

J. Pustejovsky. 1988. The geometry of events. In *Studies in Generative Approaches to Aspect: Lexicon Project Working Papers 24*, Cambridge, Massachusetts: MIT Center for Cognitive Science.

D. Radev and B. Sundheim. 2002. Using TimeML in question answering. http://www.timeml.org/site/terqas/documentation/TimeML-use-in-qa-v1.0.pdf, July.

H. Reichenbach. 1947. *Elements of Symbolic Logic*. Macmillan.

H. Reichenbach. 2005. The tenses of verbs. In J. P. Inderjeet Mani and R. Gaizauskas, editors, *The Language of Time: A Reader*, chapter 4, pages 71–78. Oxford University Press.

B. Richards. 1982. Tense, aspect and time adverbials. *Linguistics and Philosophy*, 5(1):59–107, March.

C. J. V. Rijsbergen and W. B. Croft. 1975. Document clustering: An evaluation of some experiments with the Cranfield 1400 collection. *Information Processing & Management*, 11(5-7):171–182.

C. P. Rose, B. D. Eugenio, L. S. Levin, and C. V. Ess-Dykema. 1995. Discourse processing of dialogues with multiple threads. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 31–38, Cambridge, Massachusetts, USA, June. Association for Computational Linguistics.

Z. Ruttkay. 1998. Constraint satisfaction – a survey. *CWI Quarterly*, 11(2-3):123–161.

E. Saquete and J. Pustejovsky. 2011. Automatic transformation from TIDES to TimeML annotation. *Language Resources and Evaluation*, May. doi: 10.1007/s10579-011-9147-y.

E. Saquete, P. Martínez-Barco, and R. Muñoz. 2002. Recognising and tagging temporal expressions in Spanish. In A. Setzer, editor, *Proceedings of LREC 2002 Workshop on Annotation Standards for Temporal Information in Natural Language*, pages 44–51, Las Palmas, Canary Islands, Spain, May.

E. Saquete, R. Muñoz, and P. Martínez-Barco. 2003. TERSEO: Temporal expression resolution system applied to event ordering. In *Proceedings of the 6th International Conference on Text, Speech and Dialog (TSD)*, pages 220–228, Ceske Budejovice, Czech Republic, September.

E. Saquete, P. Martínez-Barco, R. Muñoz, M. Negri, M. Speranza, and R. Sprugnoli. 2006. Multilingual extension of a temporal expression normalizer using annotated corpora. In *Proceedings of the EACL 2006 Workshop on Cross-Language Knowledge Induction*, pages 1–8, Trento, Italy, April. Association for Computational Linguistics.

E. Saquete, P. Martínez-Barco, and R. Muñoz. 2007. Evaluation of an automatic extension of temporal expression treatment to Catalan. In A. Gelbukh, editor, *Proceedings of the 8th International Conference on Computational Linguistics and Intelligent Text Processing (CICLing)*, volume 4394 of *LNCS*, pages 166–174, Mexico City, Mexico, February. Springer-Verlag.

E. Saquete, J. L. Vicedo, P. Martínez-Barco, R. Muñoz, and H. Llorens. 2009. Enhancing QA systems with complex temporal question processing capabilities. *Journal of Artificial Intelligence Research*, 35(1):755–811.

E. Saquete. 2005. *Temporal Expression Recognition and Resolution applied to Event Ordering*. Ph.D. thesis, Departamento de Lenguages y Sistemas Informaticos, Universidad de Alicante, June.

E. Saue. 2007. Eestikeelsete ajaväljendite automaatne eraldamine. Master's thesis, University of Tartu, Estonia, May.

R. Saurí, J. Littman, B. Knippen, R. Gaizauskas, A. Setzer, and J. Pustejovsky. 2006. TimeML annotation guidelines version 1.2.1, January. http://timeml.org/site/publications/timeMLdocs/annguide_1.2.1.pdf.

F. Schilder and C. Habel. 2001. From temporal expressions to temporal information: Semantic tagging of news messages. In L. Harper, I. Mani, and B. Sundheim, editors, *Proceedings of the ACL 2001 Workshop on Temporal and Spatial Information Processing*, pages 65–72, Toulouse, France, July. Association for Computational Linguistics.

F. Schilder and C. Habel. 2003. Temporal information extraction for temporal question answering. In M. T. Maybury, editor, *Proceedings of the 2003 AAAI Spring Symposium in New Directions in Question Answering*, pages 34–44, Stanford, CA, USA, March. AAAI Press.

F. Schilder, C. Habel, and Y. Versley. 2003. Temporal information extraction and question answering: Deriving answers for when-questions. In R. Bernardi and M. Moortgat, editors, *Proceedings of the 2nd CoLogNET-ElsNET Symposium –*

Questions and Answers: Theoretical and Applied Perspectives, pages 138–145, Amsterdam, The Netherlands, December.

F. Schilder. 2004. Extracting meaning from temporal nouns and temporal prepositions. *ACM Transactions on Asian Language Information Processing (TALIP)*, 3(1):33–50, March.

A. Scott. 2010. *What is Expression?: How a Formal Theory can clarify the Expressive Possibilities of Language.* iUniverse.

A. Setzer and R. Gaizauskas. 2000. Annotating events and temporal information in newswire texts. In *Proceedings of the 2nd International Conference on Language Resources and Evaluation (LREC)*, pages 1287–1294, Athens, Greece, May. European Language Resources Association.

A. Setzer. 2001. *Temporal Information in Newswire Articles: An Annotation Scheme and Corpus Study.* Ph.D. thesis, University of Sheffield, Sheffield, UK, September.

B. Shneiderman and B. B. Bederson. 2003. *The Craft of Information Visualization: Readings and Reflections.* Morgan Kaufmann Publishers Inc.

W. Siabato and M.-A. Manso-Callejo. 2011. Integration of temporal and semantic components into the geographic information through mark-up languages. part i: Definition. In B. Murgante, O. Gervasi, A. Iglesias, D. Taniar, and B. O. Apduhan, editors, *Proceedings of the International Conference on Computational Science and Its Applications (ICCSA)*, volume 6782, pages 394–409, Santander, Spain, June. Springer-Verlag.

C. S. Smith. 1978. The syntax and interpretation of temporal expressions in English. *Linguistics and Philosophy*, 2:43–99.

C. S. Smith. 1980. Temporal structures in discourse. In C. Rohrer, editor, *Time, Tense and Quantifiers*, pages 355–374. Niemeyer.

M. Stede, S. Haas, and U. Küssner. 1998. Tracking and understanding temporal descriptions in dialogue. Technical Report 232, Technische Universität Berlin, October.

J. Strötgen and M. Gertz. 2010. HeidelTime: High quality rule-based extraction and normalization of temporal expressions. In *Proceedings of the 5th International Workshop on Semantic Evaluation (SemEval)*, pages 321–324, Uppsala, Sweden, July. Association for Computational Linguistics.

J. Strötgen and M. Gertz. 2011a. Multi-lingual and cross-domain temporal tagging. *to be published.*

J. Strötgen and M. Gertz. 2011b. WikiWarsDE: A German corpus of narratives annotated with temporal expressions. In *The Proceedings of the Conference of the German Society for Computational Linguistics and Language Technology (GSCL)*, pages 129–134, Hamburg, Germany, September.

B. Sundheim and R. Grishman. 1995. Appendix C: Named entity task definition (v2.1). In *Proceedings of the Sixth Message Understanding Conference (MUC-6)*, pages 317–332, Columbia, Maryland, USA, November. Morgan Kaufmann.

C. Teissèdre, D. Battistelli, and J.-L. Minel. 2010. Resources for calendar expressions semantic tagging and temporal navigation through texts. In *Proceedings of the 7th International Conference on Language Resources and Evaluation (LREC)*, pages 3572–3577, Valletta, Malta, May. European Language Resources Association.

M. Treumuth. 2006. Automatic extraction of time expressions and representation of temporal constraints. Technical report, University of Tartu, May. http://math.ut.ee/˜treumuth/NLP/Treumuth_Term_Paper.pdf.

M. Treumuth. 2008. Normalization of temporal information in Estonian. In P. Sojka, A. Horák, I. Kopecek, and K. Pala, editors, *Proceedings of the 11th International Conference on Text, Speech and Dialogue (TSD)*, volume 5246 of *LNAI*, pages 211–218, Brno, Czech Republic, September. Springer-Verlag.

J. van Benthem. 1983. *The Logic of Time*. Reidel.

N. Vazov. 2001. A system for extraction of temporal expressions from French texts based on syntactic and semantic constraints. In L. Harper, I. Mani, and B. Sundheim, editors, *Proceedings of the ACL 2001 Workshop on Temporal and Spatial Information Processing*, pages 96–103, Toulouse, France, July. Association for Computational Linguistics.

Z. Vendler. 1957. Verbs and times. *Philosophical Review*, 56:143–160.

Z. Vendler. 1967. Verbs and times. In *Linguistics in Philosophy*, chapter 4, pages 97–121. Cornell University Press.

Z. Vendler. 2005. Verbs and times. In I. Mani, J. Pustejovsky, and R. Gaizauskas, editors, *The Language of Time: A Reader*, chapter 1, pages 21–32. Oxford University Press.

M. Verhagen, I. Mani, R. Sauri, J. Littman, R. Knippen, S. B. Jang, A. Rumshisky, J. Phillips, and J. Pustejovsky. 2005. Automating temporal annotation with TARSQI. In *Proceedings of the ACL 2005 Interactive Poster and Demonstration Sessions*, pages 81–84, Ann Arbor, Michigan, June. Association for Computational Linguistics.

M. T. Vicente-Diez, D. Samy, and P. Martinez. 2008. An empirical approach to a preliminary successful identification and resolution of temporal expressions in Spanish news corpora. In *Proceedings of the 6th International Conference on Language Resources and Evaluation (LREC)*, pages 2153–2158, Marrakech, Morocco, May. European Language Resources Association.

T. Vichayakitti and C. Jaruskulchai. 2005a. Automatic tagging time-revealing vocabulary from Thai article news. In *International Symposium on Communications and Information Technology (ISCIT)*, volume 2, pages 930–933, Beijing, China, October. Institute of Electrical and Electronics Engineers.

T. Vichayakitti and C. Jaruskulchai. 2005b. Automatic temporal event recognition from Thai news. In *International Symposium on Communications and Information Technology (ISCIT)*, volume 2, pages 938–942, Beijing, China, October. Institute of Electrical and Electronics Engineers.

M. Vilain, H. Kautz, and P. van Beek. 1990. Constraint propagation algorithms for temporal reasoning: a revised report. In D. S. Weld and J. de Kleer, editors, *Readings in Qualitative Reasoning about Physical Systems*, pages 373–381. Morgan Kaufmann Publishers Inc.

M. Vilain. 1982. A system for reasoning about time. In *Proceedings of the 2nd National Conference on Artificial Intelligence (AAAI)*, pages 197–201, Pittsburgh, Pa., USA, August. AAAI Press.

A. von Stechow. 2002. Temporal prepositional phrases with quantifiers: some additions to Pratt and Francez (2001). *Linguistics and Philosophy*, 25:755–800.

P. Waring. 2010. Human centered event linking. Master's thesis, University of Manchester.

J. Wiebe, T. P. O'Hara, T. Ohrstrom-Sandgren, and K. J. McKeever. 1998. An empirical approach to temporal reference resolution. *Journal of Artificial Intelligence Research*, 9:247–293, November.

G. Wilson, I. Mani, B. Sundheim, and L. Ferro. 2001. A multilingual approach to annotating and extracting temporal information. In L. Harper, I. Mani, and B. Sundheim, editors, *Proceedings of the ACL 2001 Workshop on Temporal and Spatial Information Processing*, pages 81–87, Toulouse, France, July. Association for Computational Linguistics.

D. Wonsever, M. Malcuori, and M. Etcheverry. 2011. Esquema de anotación de expresiones y marcas temporales. proyecto TEMANTEX. Technical Report RT 11-15, Universidad de la República, Montevideo, Uruguay, October.

M. Wu, W. Li, Q. Chen, and Q. Lu. 2005a. Normalizing Chinese temporal expressions with multi-label classification. In *Proceedings of IEEE International Conference on Natural Language Processing and Knowledge Engineering (NLP-KE)*, pages 318–323, Wuhan, China, October. Institute of Electrical and Electronics Engineers.

M. Wu, W. Li, Q. Lu, and B. Li. 2005b. CTEMP: A Chinese temporal parser for extracting and normalizing temporal information. In R. Dale, K.-F. Wong, J. Su, and O. Y. Kwong, editors, *Proceedings of the Second International Joint Conference on Natural Language Processing (IJCNLP)*, pages 694–706, Jeju Island, Korea, October. Springer-Verlag.

F. Xia and M. Palmer. 2001. Converting dependency structures to phrase structures. In *Proceedings of the First International Conference on Human Language Technology Research (HLT)*, pages 1–5, San Diego, CA, USA, March. Association for Computational Linguistics.

L. Zhou and G. Hripcsak. 2007. Temporal reasoning with medical data—a review with emphasis on medical natural language processing. *Journal of Biomedical Informatics*, 40:183–202.

Appendix A

Third-Party Software Used

In the course of preparation of this thesis we used some third-party software, either in the form of stand-alone tools or programming libraries. In this section we briefly acknowledge this software.

We used the Callisto annotation tool,[1] developed by Day et al. (2004) at MITRE, to annotate WikiWars, our corpus of Wikipedia articles about wars. It also proved to be a very useful tool for browsing annotations in other corpora, e.g. the ACE corpora, and occasionally for verifying output generated by our temporal expressions tagger, DANTE. We used Callisto because it supports various annotation schemes, including the TIMEX2 scheme which we used in our work.

The DANTE system itself is built on top of the GATE library, developed by Cunningham et al. (2002) at the University of Sheffield.[2] It provides a framework for pipeline processing, document and annotation representation and a number of processing components, called plugins, some of which (e.g. a tokeniser and a POS tagger) we used in our pipeline. The core processing components of DANTE are implemented as plugins to GATE. For example, DANTE uses the Stanford Named Entity Recogniser,[3] for which we implemented a wrapper for GATE. We also used document-editing capabilities of GATE to annotate documents for the experiments on finding the reference time for interpretation of context-dependent expressions.

For evaluation of DANTE on ACE and ACE-like datasets we used the official ACE evaluation script available from the program's website.[4] We also used the ACE converter to turn inline annotations into the ACE APF XML format.

In the experiments on extent recognition we worked with a number of syntactic parsers: Minipar,[5] Stanford,[6] Connexor,[7] C&C,[8] Bikel/Collins,[9] and the Charniak and Charniak-Johnson parsers.[10] GATE already includes wrappers for Minipar and Stanford, but we implemented our own wrappers for all the other parsers. In the experiments with these parsers we also used the Penn Treebank tokenizer,[11] which

[1] See `http://callisto.mitre.org`.
[2] See `http://gate.ac.uk`.
[3] See `http://nlp.stanford.edu/software/CRF-NER.shtml`.
[4] See `http://www.itl.nist.gov/iad/mig/tests/ace/2007/software.html`.
[5] See `http://webdocs.cs.ualberta.ca/~lindek/minipar.htm`.
[6] See `http://nlp.stanford.edu/software/lex-parser.shtml`.
[7] See `http://www.connexor.eu/technology/machinese/machinesesyntax`.
[8] See `http://svn.ask.it.usyd.edu.au/trac/candc`.
[9] See `http://www.cis.upenn.edu/~dbikel/software.html`.
[10] See `ftp://ftp.cs.brown.edu/pub/nlparser`.
[11] See `http://www.cis.upenn.edu/~treebank/tokenization.html`.

we slightly modified for our purposes. The phrase structure tree diagrams that we provide in the thesis were generated with the online phpSyntaxTree tool[12] and the GUI application available with the distribution of the Stanford parser.

We gratefully thank the authors of these tools for releasing them to the research community and for their work, which has resulted in reliable, high quality software. This is very helpful for anyone carrying out new work in these areas, since it allows one to focus on new research problems.

[12]See `http://www.ironcreek.net/phpsyntaxtree`.

Appendix B

Fixing the TIDES Parallel Corpus

In this appendix we present the modifications that were necessary for us to carry out to work with the TIDES Parallel Corpus annotated with TIMEX2. A description of this corpus is provided in Section 2.5.2 of this thesis (see page 45).

The corpus is distributed as a single file containing all the documents that make up the corpus. A simple search over this file reveals that there are 7306 opening tags for `TIMEX2` elements, but only 7302 closing tags. To make it easier to fix the corpus, we first split it into a form where each document is a separate file; this resulted in 95 files. These files still have content in two languages: Spanish and English.

The problem with matched opening and closing tags occurred in seven documents, with one error per document. The types of errors were different:

- **an opening instead of a closing tag** was used in documents W5_0005 and E_0033:

  ```
  bueno, entonces, te veo <TIMEX2 VAL="1999-08-09">el lunes<TIMEX2>,
  ```

  ```
  That is any schedule whether it is in <TIMEX2 VAL="1999-05-19-TMO">the
  morning<TIMEX2> or at <TIMEX2 VAL="1999-05-19-T12">noon</TIMEX2>
  ```

- **a closing instead of an opening tag** was used in document W12_0005:

  ```
  I think that the best would be to meet during <TIMEX2 VAL="1999-W23">the
  week of </TIMEX2>the 6th</TIMEX2></TIMEX2>
  ```

- **the name of the closing tag was misspelled** in documents W5_0006 and W5_0011

  ```
  y luego nos juntamos de <TIMEX2 VAL="1999-XX-16T13">una</TIME2>
  ```

  ```
  entre <TIMEX2 VAL="1999-03-08T11">las once</TIME2>
  ```

- **a closing tag was missing** in W5_0009,

  ```
  estoy libre en <TIMEX2 VAL="1999-04-09TAF">la tarde
  ```

- **an opening tag was missing** in E_0008:

  ```
  So, is it inconvenient to have it on
  <TIMEX2 VAL="1999-06-09TAF">the ninth</TIMEX2> In the afternoon</TIMEX2>?
  ```

  As explained in the TIMEX2 2001 Annotation Manual (Ferro, 2001, pp. 63–64) there should be an opening tag before *In the afternoon* (which also should not start with the uppercase I), and consequently the value of the first temporal expression should be `1999-06-09`. Tagging such expressions as two temporal expressions without embedding is the approach still taken in the most recent version of TIMEX2 from 2005.

After fixing the above issues there were 7306 TIMEX2 annotations. However, there were still a number of problems which meant that 14 documents in the corpus were not parseable as XML documents. These problems were as follows:

- **missing opening quotes before a value of the `VAL` attribute** in document E_0002
- **missing closing quotes after a value of the `VAL` attribute** in documents E_0009 and E_0015:

  `after <TIMEX2 VAL="XXXX-XX-14T13:30>one thirty</TIMEX2>`
- **an incorrect value provided for the `VAL` attribute** in documents W11_0003, W11_0005 and W11_0006, where ampersand characters were not encoded as XML entities:

  `<TIMEX2 VAL="XXXX-WXX & XXXX-WXX">these next two weeks</TIMEX2>`
- **incorrect assignment of a value of the `VAL` attribute to the name of the XML element** in documents E_0025 and W5_0006:

  `until <TIMEX2="XXXX-XX-XX-T13">one</TIMEX2>.`
- **missing closing quotes after a value of the `MOD` attribute** in documents W5_0013 (two errors), E_0001 (in this case the value was empty, so the whole attribute can be dropped), and W12_0016.
- **missing closing quotes for the value of the `GRANULARITY` attribute** in document E_0021
- **missing opening quotes for the value of the `PERIODICITY` attribute** in document W5_0007
- **missing the '>' character in the element tag** in document W12_0002:

  `a <TIMEX2 VAL="T17" SET="YES" PERIODICITY="F1D"cinco</TIMEX2>`

Also, in E_0015 an expression was annotated twice, so one of the annotations was spurious and we removed it.

Fixing these issues resulted in 7305 TIMEX2 annotations, 3541 of which (48.47%) occurred in the English part of the corpus, and made the corpus useable as an XML resource.

We note that there were also issues related to values provided in formats not compatible with the TIMEX2 guidelines, although being allowable XML attribute values. We already mentioned the values of the `VAL` attributes containing the ampersand; similarly, the document E_0015 contained five annotations with the slash character, as in the following example:

(B.1) `<TIMEX2 VAL="XXXX-XX-28/XXXX-XX-30">The last three days of this month`

There were also 137 annotations whose `VAL` attributes incorrectly started with a space character, and one annotation in document W5_0008 with an incorrectly encoded time:

(B.2) `<TIMEX2 VAL="1999-WXX-1T1:30">one thirty</TIMEX2>`

Appendix C

Evaluation Metrics

Precision, Recall, and F-Measure

Precision and recall are defined as in Definition C.1.

Definition C.1 *Let A denote the set of all temporal expressions in the text and let B denote all phrases in the text marked by the tagging system as temporal expressions. Then we define precision as:*

$$P = \frac{|A \cap B|}{|B|} \tag{C.1}$$

and recall as:

$$R = \frac{|A \cap B|}{|A|}. \tag{C.2}$$

Less formally we can describe P and R as:

$$P = \frac{\text{the number of correct answers provided by the system}}{\text{the number of all answers provided by the system}}$$

$$R = \frac{\text{the number of correct answers provided by the system}}{\text{the number of temporal expressions in the text}}$$

F-measure is a measure combining precision and recall. Originally derived by Rijsbergen and Croft (1975) to measure the effectiveness of information retrieval systems, it has been defined as follows:

$$F_\alpha = \frac{RP}{(1-\alpha)P + \alpha R}, \quad 0 \le \alpha \le 1. \tag{C.3}$$

Another variant of the above equation is often found in the literature, substituting α with β, where $\alpha = 1/(\beta^2 + 1)$. Then we get:

$$F_\beta = \frac{(\beta^2 + 1)RP}{\beta^2 P + R}, \quad 0 \le \beta \le \infty. \tag{C.4}$$

If $\beta = 1$ we have F_1 (also called the balanced F-measure), which equally weights precision and recall and is the harmonic mean of these two quantities:

$$F_1 = \frac{2RP}{P + R}. \tag{C.5}$$

This is the most often used F-measure in evaluations of the TERN task. If we set $\beta = 2$ then F_2 weights recall twice as much as precision, and if $\beta = 0.5$ then $F_{0.5}$ weights precision twice as much as recall.

Reports of precision, recall and F-measure in the literature are sometimes backed-up with four counts and two variables used to calculate precision and recall:

- *CORR* (correct): the number of items generated by the system and provided in the answer key that are considered to be identical.
- *INCO* (incorrect): the number of items generated by the system and provided in the answer key that are not considered to be identical. Sometimes these are also referred as *partially correct*.
- *MISS* (missing): the number of items in the answer key that are not generated by the system.
- *SPUR* (spurious): the number of items generated by the system that are not marked in the answer key. These are also referred as *false positive* items.
- *ACT* (actual): the total number of items generated by the system: $\text{ACT} = \text{CORR} + \text{INCO} + \text{SPUR}$.
- *POSS* (possible): the total number of items in the answer keys: $\text{POSS} = \text{CORR} + \text{INCO} + \text{MISS}$.

Then the calculation of precision and recall values can be carried out with the following formulas:

$$\text{P}_{str} = \frac{\text{CORR}}{\text{ACT}} = \frac{\text{CORR}}{\text{CORR} + \text{INCO} + \text{SPUR}} \tag{C.6}$$

$$\text{R}_{str} = \frac{\text{CORR}}{\text{POSS}} = \frac{\text{CORR}}{\text{CORR} + \text{INCO} + \text{MISS}} \tag{C.7}$$

The above metrics of precision and recall are called *strict*, because they only consider fully matched expressions.[1] As such the strict metrics are used to evaluate the recognition task (where the exact extents are to be found). There are also *lenient* precision and recall measures defined, which treat correctly and partially-correctly-recognized expressions in the same way:

$$\text{P}_{len} = \frac{\text{CORR} + \text{INCO}}{\text{ACT}} = \frac{\text{CORR} + \text{INCO}}{\text{CORR} + \text{INCO} + \text{SPUR}} \tag{C.8}$$

$$\text{R}_{len} = \frac{\text{CORR} + \text{INCO}}{\text{POSS}} = \frac{\text{CORR} + \text{INCO}}{\text{CORR} + \text{INCO} + \text{MISS}} \tag{C.9}$$

These metrics are used to evaluate the detection tasks, where an expression is counted as a positive outcome even when it is not recognized correctly but only partially.

In the literature, the *average* of strict and lenient measures are also sometimes used; these can be computed with the following formulas:

$$\text{P}_{av} = \frac{\text{CORR} + 0.5 \cdot \text{INCO}}{\text{ACT}} = \frac{\text{CORR} + 0.5 \cdot \text{INCO}}{\text{CORR} + \text{INCO} + \text{SPUR}} \tag{C.10}$$

[1]We note that the strict precision metric penalises partially correct answers in favour of missing ones, which is counterintuitive (it would seem that it is more useful to find at least a part of a temporal expression than nothing at all).

$$\mathrm{R}_{av} = \frac{\mathrm{CORR} + 0.5 \cdot \mathrm{INCO}}{\mathrm{POSS}} = \frac{\mathrm{CORR} + 0.5 \cdot \mathrm{INCO}}{\mathrm{CORR} + \mathrm{INCO} + \mathrm{MISS}} \tag{C.11}$$

Depending on which type of precision and recall is used, the F-measure is then also called *strict*, *lenient* or *average*.

Accuracy

Accuracy is defined as in Definition C.2.

Definition C.2 *For a set of n expressions which a system has correctly identified in the text, let m be the number of correct values of a specific attribute generated by the system. Then we define the accuracy of the system for this attribute as: $Acc = \frac{m}{n}$.*

Less formally, the accuracy is the number of correct answers provided by the system divided by the total number of values generated by the system.

ACE 2004 Value

At TERN 2004, a metric called the *value score* was introduced. This is a weighted measure which allows taking account of the perceived relative importance of different attributes of the annotations. The overall score for a system is the sum of scores for all generated annotations associated with temporal expressions:

$$System\_Score = \sum_i Value(sys\_timex_i) \tag{C.12}$$

The system gets no score for missing a temporal expression. The value for a generated temporal expression is defined as a product of two factors:

$$Value(sys\_timex_i) = Timex\_Value(sys\_timex_i) \cdot \prod_k W_{Terr\text{-}k}(sys\_attr_k) \tag{C.13}$$

The first factor accumulates the score for the generated annotation by summing attribute weights (see Table C.1) for finding the expression (detection) and those attributes that also exist in the corresponding gold-standard annotation:

$$Timex\_Value(timex) = \begin{cases} \sum_{i \in sys\_attr} Attr\_Value(i) & \text{if matched} \\ \sum_{i \in sys\_attr} AttrWeight(i) & \text{if spurious} \end{cases} \tag{C.14}$$

where

$$Attr\_Value(attr) = \begin{cases} AttrWeight(attr) & \text{if reference attribute exists} \\ 0 & \text{otherwise} \end{cases} \tag{C.15}$$

If the generated annotation is not contained in the gold standard (i.e. the system generated a spurious annotation), then the value score is the sum of the weight for detection and weights for all the generated attributes. The annotations are considered to be matched if they have at least one character overlap.[2]

[2]The official evaluation guidelines (see `http://www.itl.nist.gov/iad/mig/tests/ace/2004`) do not mention what overlap was considered, but the system descriptions published by Hacioglu et al. (2005) and Negri and Marseglia (2005) reveal that it was one character.

Attribute	val	mod	set	anchor_val	anchor_dir	detection
Weight	1.00	0.10	0.10	0.50	0.25	0.10

Table C.1: Attribute weights used in the TERN 2004, 2005 and 2007 evaluations.

Discount weight	MinScore	Hard	Default	Easy	MaxScore
$W_{\textit{Terr-ANCHOR\_DIR}}$	0.00	0.50	0.75	0.90	1.00
$W_{\textit{Terr-ANCHOR\_VAL}}$	0.00	0.00	0.50	0.75	1.00
$W_{\textit{Terr-MOD}}$	0.00	0.50	0.90	1.00	1.00
$W_{\textit{Terr-SET}}$	0.00	0.50	0.75	0.90	1.00
$W_{\textit{Terr-VAL}}$	0.00	0.00	0.25	0.50	1.00
$W_{\textit{T-FA}}$	1.00	1.00	0.75	0.50	0.00

Table C.2: Discount weights used in the TERN 2004 evaluations.

The second factor is a dynamic coefficient which penalises incorrect attribute values generated by the system. Each discount weight is in the range $[0; 1]$, as shown in Table C.2; by multiplying the score of the annotation by these weights, the score is lowered. The evaluation assumed three levels of difficulty, by defining three sets of weights; in the 'hard' setting, the penalties were most severe. For generating a spurious annotation, the score of the annotation was multiplied by the false alarm weight; as the score for a spurious annotation is subtracted from the overall score of the system, here the hard setting has the highest value for the weight (so a larger number is subtracted).[3]

ACE 2005 Value

At TERN 2005, the evaluation metric was changed from that used in ACE 2004, becoming normalized by the score for reference (gold-standard) answers:

$$\textit{System\_Score} = \frac{\sum_i \textit{value\_of\_sys\_token}_i}{\sum_j \textit{value\_of\_ref\_token}_j} \tag{C.16}$$

The result is given as a percentage, with the maximum value being 100%, but as the system is given negative points for spurious expressions and attributes, the overall result can also be negative. The value for one TE is calculated using the following formula:

$$\textit{Value}(\textit{timex}) = \textit{Element\_Value}(\textit{timex}) \cdot \textit{Mentions\_Value}(\textit{timex}) \tag{C.17}$$

The *Element_Value* factor indicates how well the attributes of a system-produced annotation match the attributes of the corresponding reference annotation:

$$\textit{Element\_Value}(\textit{timex}) = \begin{cases} \sum_{i \in \textit{sys\_Attr}} \textit{Attr\_Value}(i) & \text{if matched} \\ \sum_{i \in \textit{sys\_Attr}} \textit{AttrWeight}(i) \cdot W_{FA} & \text{if spurious} \end{cases} \tag{C.18}$$

[3]The documentation of the ACE 2004 evaluations, which is the source for formulas (C.12)–(C.15) and Table C.2, does not specify the subtraction operation, but an investigation of the source code of the evaluation script reveals that this is what is done to the score. Also, private communication with NIST confirms that the formulas in the documentation may have contained a mistake.

where

$$Attr\_Value(attr) = \begin{cases} AttrWeight(attr) & \text{if sys. and ref. attributes match} \\ 0 & \text{otherwise} \end{cases} \tag{C.19}$$

The attribute weights remained unchanged from ACE 2004 (see Table C.1). Generating a spurious annotation results in subtracting the score value of this annotation multiplied by the W_{FA} discount weight, which was set to 0.75.

The second factor in *Value*(*timex*), i.e. *Mentions_Value*(*timex*), is the sum of the mutual mention value (MMV) for all mentions of the temporal expression:[4]

$$Mentions\_Value(timex) = \sum_{\substack{all\\docs}} \sum_{\substack{all\ sys\\mentions\\in\ doc}} MMV(mention_{sys}) \tag{C.20}$$

where

$$MMV(mention_{sys}) = \begin{cases} 1 & \text{if } mention_{sys} \text{ mapped} \\ -(W_{M-FA} \cdot W_{M-CR}) & \text{if spurious} \end{cases} \tag{C.21}$$

The idea is that for system annotations matched with the gold standard, the *MMV*() factor does not modify the score calculated with *Element_Value*(); for spurious annotations the score is multiplied by a negative weight, which lowers the overall score of the system.[5]

System mentions and reference mentions are treated as matched if their extents have a mutual overlap, measured as a ratio using formula (C.22), higher than *min_overlap*=0.3.

$$mutual\_overlap = \frac{sys\_extent \cap ref\_extent}{\max(sys\_extent,\ ref\_extent)} \tag{C.22}$$

ACE 2007 Value

Conceptually, the idea of the ACE 2007 score value is the same as in ACE 2005. There are differences in the formulas presented in the corresponding evaluation plans, but the mechanism for calculating the score remained the same; also the attribute weights were the same (see Table C.1). The scoring scripts are however slightly different, and therefore it is not possible to directly compare the results from the two evaluations; our experiments with the two scorers showed that the ACE 2005 scorer was more lenient with the default parameter settings than the 2007 scorer.[6]

[4]The prepared evaluation formulas are an overkill here because in the TERN task each TE can have only one mention (unlike, for example, in the entity recognition task, where an entity can have numerous mentions in the text).

[5]The official evaluation plan for ACE 2005 specifies $W_{M-FA} = 0.75$ and $W_{M-CR} = 0.00$, which would mean that for spurious annotations the overall system gets zero points, not a negative score. Our experiments with the scoring script revealed that the results change with setting different values to W_{M-FA}, which indicates that $W_{M-CR} \neq 0.00$. Our communication with NIST confirmed the issues with the evaluation; however, we have not been able to resolve the issue with the W_{M-CR} constant.

[6]We found that changing the value of the W_{M-FA} constant from the default 0.75 to 1 gives the same results as the ACE 2007 scorer, which does not use this constant at all. Our communication with NIST confirmed the observed differences regarding the W_{M-FA} constant and we were told that there were tweaks to parameter settings and possible bug fixes between 2005 and 2007. It also turned out that at some point the 2005 systems were rescored with the 2007 code.