INSTITUTO SUPERIOR TÉCNICO
Universidade Técnica de Lisboa

Machine Learning Methods for Resolving Temporal References in Text

Vítor Sérgio Santos Loureiro
(Bachelor in Information Systems and Computer Engineering)

Dissertation for the achievement of the degree:

Master in Information Systems and Computer Engineering

Committee

Chairman:	Prof. Doutor Luis Rodrigues
Main supervisor:	Prof. Doutor Bruno Martins
Co supervisor:	Prof. Doutor Pável Calado
Observers:	Prof. Doutor Nuno Mamede

November 2011

Abstract

Time expressions, which are used to elucidate the changes of the world, are crucial in many of the natural language processing (NLP) applications, such as question answering, text summarization, temporal information retrieval systems, etc.

This thesis presents a machine learning method for resolving temporal references in text, i.e. linking particular character strings in documents to the corresponding time instances and intervals. This is a fundamental task for Temporal Information Retrieval, supporting the access through time to large document collections.

The proposed method is an instance of stacked learning, in which a first learner based on Conditional Random Fields is used to tag temporal references in the text, and then a second learner based on SVM regression is used to rank and choose from a set of possible candidate disambiguations for the temporal references that were initially tagged. The proposed method was evaluated on English corpora containing TIMEX2 and TimeML annotations for the temporal references.

The best results, in terms of F_1 measure, that the proposed method achieved were 0.93 and 0.67 respectively for recognition and normalization tasks, which shows that the proposed method compares well against previously proposed approaches.

Keywords: Temporal Expressions , Machine Learning , Information retrieval

Resumo

As expressões temporais, usadas para nos elucidar das mudanças no mundo, são cruciais em várias aplicações de processamento de lingua natural, tais como, sistemas de pergunta-resposta, sumarização de texto, sistemas de extracção de informação temporal, etc.

Nesta tese é apresentado um método de aprendizagem automática para a resolução de expressões temporais, isto é, ligar sequencias de caracteres nos documentos aos correspondentes intervalos e instancias de tempo. Esta é uma tarefa fundamental na extracção de informação temporal, suportando o acesso a grandes quantidades de documentos através do tempo.

O método proposto é uma instancia da chamada aprendizagem em pilha, na qual um primeiro modelo de aprendizagem baseado em Conditional Random Fields é usado para etiquetar referencias temporais no texto e um segundo modelo de aprendizagem baseado em regressão por SVM é usado para classificar e escolher entre um conjunto de desambiguações possível para a referencia temporal inicialmente etiquetada.

O método proposto foi avaliado usando um conjunto de dados em Inglês contendo anotações TIMEX2 e TimeML para as referências temporais.

Os melhores resultados, em termos de medida F_1, que o método proposto alcançou foram 0.93 e 0.67 para as tarefas de reconhecimento e desambiguação respectivamente, o que mostra que esta abordagem se compara bem contra abordagens propostas anteriormente.

Keywords: Expressões Temporais , Aprendizagem Automática , Extracção de Informação

Agradecimentos

A realização deste trabalho representa para mim o culminar de um ciclo de aprendizagem que não teria sido possível sem o apoio de várias pessoas às quais expresso, desde já, a minha gratidão.

Em primeiro lugar, o meu sincero obrigado à minha família, em particular aos meus pais que sempre me proporcionaram tudo o que precisei para completar com sucesso esta importante etapa da minha vida. Um obrigado muito especial para ti também, Marina, por todo o apoio, carinho e motivação.

Por outro lado, quero agradecer a todos os meus colegas e amigos que me acompanharam no meu percurso académico e que muito apoio me deram ao longo destes anos, ajudando-me a perceber que o companheirismo e a entreajuda são essenciais, quer na vida académica, quer na profissional. Estou certo que durante este período ganhei amigos para a vida! Um abraço especial ao Erik Wennberg e ao Marcos Simões, que me aturaram como companheiro de grupo.

Como não poderia deixar de ser, quero agradecer aos meus incansáveis orientadores, o Prof. Pável Calado e o Prof. Bruno Martins, pela grande dedicação e apoio, contribuindo, indubitavelmente, para o meu progresso, empenho e proveito no presente estudo. Em particular, quero deixar um agradecimento especial ao Prof. Bruno Martins, pela confiança dada ao incorporar-me no projecto SInteliGIS suportado pela Fundação para a Ciência e Tecnologia (FCT).

Por fim, gostaria de estender os meus agradecimentos a todos os colegas do grupo DMIR que sempre se revelaram empenhados para me esclarecerem quando as dúvidas me assolavam e, também, pelas ideias fornecidas que muito me valeram para completar com mérito mais um ciclo da minha vida.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Temporal information is pervasive over textual documents, since most of them contain references to particular calendar dates, clock times or duration periods. An important text analytics problem is therefore related to resolving these temporal references, i.e. linking the character strings in the documents that correspond to temporal references to the time intervals that they refer to. However, temporal reference resolution presents several non-trivial problems, due to the inherent ambiguity and contextual assumptions of natural language discourse.

Over the last few years, the temporal reference resolution problem has been addressed by many different researchers (Ahn *et al.*, 2007; Mani & Wilson, 2000; Verhagen & Moszkowicz, 2009). The problem is generally divided into two separate sub-tasks, namely (i) temporal reference identification, and (ii) temporal reference disambiguation. The first sub-task is deeply related to the problem of Named Entity Recognition (NER), which has been thoroughly studied in the natural language processing (NLP) community (N. & Sekine, 2007; Sang & Meulder, 2003). The second sub-task involves re-expressing the identified temporal references into a standard format which precisely describes their semantics.

Traditional methods for addressing the temporal reference resolution problem are based on rules and heuristics hand-tuned by experts, which are not particularly robust or generalizable. This is particularly true for the disambiguation subtask, where rules must combine knowledge about how the various temporal primitives are related (e.g., a day corresponds to 24 hours, a week corresponds to 7 days, February in a leap year has 29 days, etc.) together with information about how temporal primitives interact with one another, given a description manifested by a temporal expression. Thus, a particularly interesting challenge relates to the effective application of data-driven methods in the two tasks involved in the temporal reference resolution problem, using principled approaches for combining different sources of evidence available from training data.

While machine learning methods are already widely used in the recognition subtask, very few studies have addressed their application to the disambiguation subtask.

This thesis proposes a supervised machine learning method for resolving temporal references in text. The proposed method is an instance of stacked learning (Wolpert, 1992), in which a first learner based on Conditional Random Fields (CRF) is used to recognize and classify temporal references, and then a second learner based on Support Vector Machine (SVM) regression is used to rank the possible candidate disambiguations for the temporal references that were initially tagged. In terms of the implementation, the first learner is based on the CRF tagger from the LingPipe package (Carpenter & Baldwin, 2011). The second learner is based on the SVM regression implementation available in the Weka machine learning toolkit (Witten & Frank, 2000), using a small set of rules for generating the candidate disambiguations. Different configurations of the proposed method (e.g., different feature sets) were evaluated through English corpora containing TIMEX2 and TimeML annotations for the temporal references. Results show that machine learning methods are indeed effective and that they compare well against previously proposed state-of-the-art approaches.

1.1 Contributions

The main objective of this research was to introduce a novel approach to the temporal expressions resolution problem, machine learning techniques for both recognition and disambiguation tasks. This main objective can be divided in three important sub-goals, which correspond to the following main contributions for this problem:

- The recognition of temporal references can be addressed effectively through the formation of Conditional Random Fields (CRFs). A CRF model using a rich set of features obtained recognition F1 measure between 0.64 and 0.93, over TIMEX2 and TimeML annotated datasets well known in the area.

- The development of a module based on rules, which generates several candidates to resolve a temporal expression, given a temporal expression. This module can produce thousands of candidates as it keeps track of all the temporal expressions present in the document and its respectives candidates, and use them to generate candidates for the current temporal expression being computed.

- In the recent years some effort has been spent regarding the usage of machine learning in the temporal expressions normalization task. In this research a machine learning method

using Support Vector Machine (SVM) regression with a rich set of features obtained F1 values between 0.31 and 0.67, over the same datasets used for recognition.

The work described in this dissertation was also partially described in the following publications:

- In (Loureiro *et al.*, 2011b) we presented the supervised machine learning method for resolving temporal references in text that I present in this thesis. In this paper we use a first model based on Conditional Random Fields (CRFs) and a second learner based on SVM regression is used to rank the possible candidate disambiguations for the temporal references that were initially tagged. The proposed method was evaluated through English corpora containing TIMEX2 and TimeML annotations.
- In (Loureiro *et al.*, 2011a) we propose a supervised learning method for resolving geo-temporal references in text, using a pair of models based on Conditional Random Fields (CRF) to recognize and classify the geo-temporal references, and a second pair of models based on Support Vector Machine (SVM) regression to rank the possible candidate disambiguations for the references that were initially tagged. The proposed method was evaluated through English corpora containing SpatialML annotations for the place references, or TIMEX2 and TimeML annotations for the temporal references.

1.2 Organization

The rest of this document is organized as follows,

- Chapter 2 presents the fundamental concepts required for understanding the contents of this dissertation, introducing machine learning methods such as Hidden Markov Models (HMM) and Conditional Random Fields (CRF).
- Chapter 3 presents related work, covering both the areas of named entity recognition and temporal reference resolution. In particular, Chapter 3 details some of the previously proposed systems that address the recognition and normalization of temporal expressions, ending with a comparative analysis.
- Chapter 4 presents the proposed stacking-based machine learning approach for the resolution of temporal references, detailing the recognition and disambiguation models.
- Chapter 5 presents the experimental validation of the proposed machine learning approach, detailing both the experimental methodology and the obtained results.

- Finally, Chapter 6 summarizes the main conclusions and points directions for future work in the area.

Chapter 2

Concepts

Resolving temporal references in text is generally divided into two separate sub-tasks, namely (i) temporal reference recognition and (ii) temporal reference normalization (or disambiguation).

The recognition task aims to delimit character strings corresponding to a set of different types of temporal expressions. These types can either refer to points in time (e.g., *2 June 1986*), durations (e.g., *6 seconds*) or recurring events (e.g., *every 4 hours*). Recognizing temporal expressions in text is a particular aspect of the more general problem of Named Entity Recognition (NER), that in turn is a subtask of Information Extraction that seeks to locate and classify atomic elements in text into predefined entity categories (e.g., names of persons, organizations, locations, expressions of time or numeric quantities) (Maynard *et al.*, 2001; N. & Sekine, 2007). Named Entity Recognition problems have been studied along the years by many different researchers and with many different methods.

The temporal expression normalization task aims to interpret the recognized expressions in order to represent them in a standard format, for example through associations to dates or intervals in a calendar. The normalization task is considerably more challenging, since the interpretation of *timexes* requires some sort of reasoning over the data for relating each *timex* to other *timexes* given in the same document (N. & Sekine, 2007). Moreover, the normalization step represent *timexes* in a way that facilitates further processing (Han, 2003; Pustejovsky *et al.*, 2003; Wilson *et al.*, 2001). Normalization is also less suitable to machine learning methods.

Section 2.1 will detail named entity recognition methods, particularly Hidden Markov Models and Conditional Random Fields, section 2.2 will give an overview of the evaluation methods used for entity recognition, Section 2.3 will detail well known annotation schemes for temporal expressions, and finally Section 2.4 will summarize this Chapter.

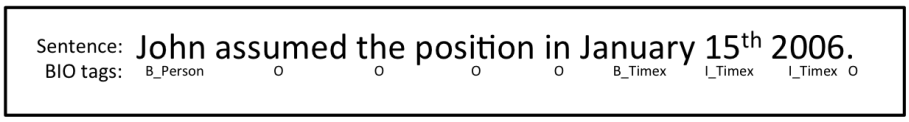

Figure 2.1: BIO encoding example

2.1 Named Entity Recognition

The named entity recognition (NER) task, as proposed in the natural language processing (NLP) community, refers to identifying and classifying atomic elements in text into predefined categories, such as the names of persons, organizations, locations, expressions of times, quantities, monetary values, percentages, etc. Current state-of-the-art systems can achieve near-human performance values, effectively handling the ambiguous cases (e.g., the same proper name can refer to distinct real world entities) and achieving F_1 scores around the mark of 90%. N. & Sekine (2007) provide a recent survey in the area.

The NER task can be addressed through two main types of sequence-modeling techniques, namely (i) *token-level* and (ii) *segment-level* techniques. The first type of techniques, and the one that is most often used, treats the text as a sequence of tokens and the extraction problem as the assignment of an entity label to each token. Since entities typically comprise multiple tokens, it is custom to decompose each entity label as *"Entity Begin"*, *"Entity End"*, and *"Entity Continue"*. This is popularly known as the BCEO (where B=Begin, C=Continue, E=End and O=Other) encoding. Another popular encoding is BIO, decomposing an entity label into *"Entity Begin"*, *"Entity Inside"* and *"Entity Other"*. In the segment-level sequence-modeling technique the output is a sequence of segments, with each segment defining an entity, rather than a sequence of labels as in token-level models. Figure 2.1 shows an example of the BIO tag encoding.

Initial NER approaches, which are nonetheless still commonly used, were based on manually constructed finite state patterns and/or dictionary lists of entity names. In general, the pattern-based approaches attempt to match sequences of words in much the same way as a general regular expression matcher (Aho & Corasick, 1975). Despite being robust and capable of achieving state-of-the-art performances, these initial methods lack the ability of coping with the problems of robustness and portability. Each new source of text requires significant tweaking of rules and/or dictionaries to maintain optimal results.

The current trend in NER is to use machine-learning approaches, relying on features extracted from training data that reflect properties of (i) individual named entities (e.g., capitalization, character types, entity type and frequency) and (ii) general statistical measures either at the document

scale or at the corpus scale. Machine learning approaches are more attractive in that they are trainable and adaptable, at the same time maintaining state-of-the-art performance results. Several different classification methods have been successful applied on this task (Sang & Meulder, 2003). For instance Mayfield *et al.* (2003) applied Support Vector Machines (SVM) to classify each individual name entity. Chieu & Ng (2003) and Bender *et al.* (2003) applied Maximum Entropy (ME) approaches, using local features occurring near individual tokens and global features from the whole document. Zhou & Su (2002) experimented with Hidden Markov Models (HMMs), also using a large variety of features. Conditional Random Fields (CRFs), a generalization of ME and HMMs, were explored by McCallum & Li (2003). Florian *et al.* (2003) combined Maximum Entropy and Hidden Markov Models under different conditions. Recent works have shown that CRF models have advantages over traditional HMMs in the face of many redundant features, although CRF models need fairly extensive feature engineering in order to outperform a good HMM baseline. This dissertation describes Hidden Markov Models (HMMs) and Conditional Random Fields in the next two sub-sections.

2.1.1 Hidden Markov Models

Hidden Markov modeling is a powerful statistical machine learning technique used in many information extraction tasks. These models have well understood statistical foundations that can be extrapolated to natural language domains, handling new data robustly. This approach, like most of the statistical methods for entity extraction, is based on token-level sequence-modeling.

Figure 2.2 shows the general architecture of an instantiated HMM. Each oval shape represents a random variable that can adopt any of a number of values. The random variable Y_i is the hidden state at sequential position i (i.e., the label for the token at position i). The random variable X_i is the observation at sequential position i (i.e., the word token at position i) of the input token sequence. The arrows in the diagram denote conditional dependencies.

As we can see from the diagram, the conditional probability distribution of the hidden variable Y at sequential position i, depends only on the value of the hidden variable Y_{i-1}. The values at $i-2$ and before are not considered and do not have a direct influence. This is called the Markov assumption. Similarly, the value of the observed variable X at sequential position i only depends on the value of the hidden variable Y at sequential position i.

Now, let us denote $A = P(Y_{i+1}|Y_i)$, i.e., the probability of being in state Y at time $i+1$, given that we were in state Y at time i, $B = P(X_i|Y_i)$, i.e., the probability of observing the symbol X_i, given that we are at state Y at time i, and $\pi = P(X_1)$, i.e., the probability of being in state X at the beginning of the experiment. With this notation, we can denote an HMM as $\lambda = (A, B, \pi)$.

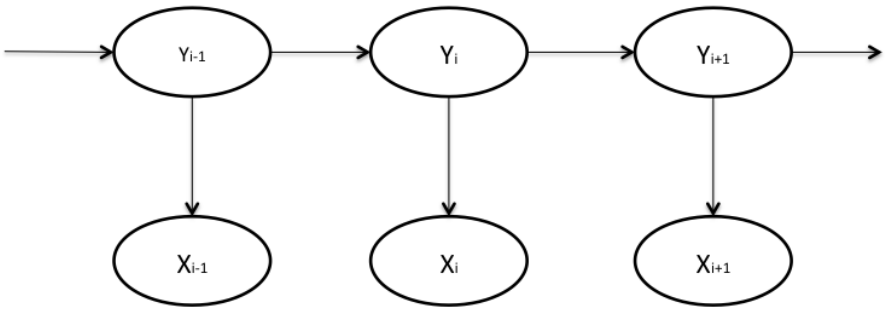

Figure 2.2: Conceptual architecture of Hidden Markov Models

Now, our problem is to find a state sequence $Y = Y_1, Y_2, ..., Y_t$ such that the probability of the observation sequence $X = X_1, X_2, ..., X_t$ having been generated from this sequence Y is greater than that from any other state sequence, i.e., find the Y that will maximize $P(X, Y|\lambda)$. This type of problem can be solved efficiently using the Viterbi algorithm (Dugad & Desai, 1996; Viterbi, 1967). The Viterbi algorithm is a dynamic programming method that computes the combination of states that maximizes the above probability, so that in the final step of the algorithm we have the most likely sequence of states that could derive the observed sequence.

The parameters $\lambda = (A, B, \pi)$ involved in the HMM have to be learned from a set of examples. The task usually involves deriving the maximum likelihood estimate of the parameters of the HMM, given a set of output sequences. This maximum likelihood estimate can be derived efficiently using the Baum-Welch algorithm, a particular case of the more general expectation-maximization (EM) algorithm, or the Segmental K-means algorithm. The Baum-Welch algorithm makes use of dynamic programming to efficiently adjust the parameters of the model so as to increase $P(X|\lambda)$ until a maximum value is reached (Baum *et al.*, 1970).

There are several implementations of HMMs and among the most well known are the ones provided by the LingPipe[1] and Mallet[2] packages.

2.1.2 Conditional Random Fields

Conditional random fields (CRFs) are a probabilistic framework for labeling and segmenting sequential data, based on a conditional approach. A CRF is an undirected graphical model that

[1] http://alias-i.com/lingpipe/
[2] http://mallet.cs.umass.edu/

defines a single log-linear distribution over label sequences, given a particular observation sequence. The primary advantage of CRFs over HMMs is their conditional nature, resulting in the relaxation of the independence assumptions required by HMMs in order to ensure tractable inference (Lafferty *et al.*, 2001).

The probability of a particular sequence of labels Y_i given the sequence of observations X_i is a normalized product of potential functions, each of form:

$$P(Y_i, X_i) = \exp\left(\sum_j \lambda_j t_j\left(Y_{i-1}, Y_i, X, i\right) + \sum_k \mu_k s_k\left(Y_i, X, i\right)\right)$$

In the formula, $t_j\left(Y_{i-1}, Y_i, X, i\right)$ is a transition feature function of the entire observation sequence and the labels at positions i and $i-1$ in the label sequence, $s_k\left(Y_i, X, i\right)$ is a state feature function of the label at position i and the observation sequence, and j and μ_k are parameters to be estimated from training data. When defining feature functions, we construct a set of real-valued features $b(X, i)$ of the observation, to express some characteristic of the empirical distribution of the training data that should also hold for the model distribution. An example of such a feature is:

$$b\left(X, i\right) = \begin{cases} 1 & \text{if } the\ observation\ at\ position\ i\ is\ the\ word\ "Monday" \\ 0 & otherwise \end{cases}$$

Each feature function takes on the value of one of these real-valued observation features $b(X, i)$ if the current state (in the case of a state function) or previous and current states (in the case of a transition function) take on particular values. All feature functions are therefore real-valued. For example, consider the following transition function:

$$b\left(X, i\right) = \begin{cases} b\left(X, i\right) & \text{if } Y_{i-1} = NOUN\ and\ Y_i = VERB \\ 0 & otherwise \end{cases}$$

For notation purposes, we simplify the equation above by writing:

$$s\left(Y_i, X, i\right) = s\left(Y_{i-1}, Y_i, X, i\right)$$

$$F_j\left(Y, X\right) = \sum_{i=1}^{n} f_j\left(Y_{i-1}, Y_i, X, i\right)$$

In the formula, each $f_j\left(Y_{i-1}, Y_i, X, i\right)$ is either a state function $s\left(Y_{i-1}, Y_i, X, i\right)$ or a transition function $t\left(Y_{i-1}, Y_i, X, i\right)$. This allows the probability of a label sequence Y, given an observation

sequence X, to be written as:

$$p\left(Y|X,\lambda\right)=\frac{1}{Z\left(X\right)}\exp\left(\sum_{j}\lambda_{j}F_{j}\left(Y,X\right)\right)$$

In the formula, $Z\left(X\right)$ is a normalization factor over all state sequences, $f_{j}\left(Y_{i-1},Y_{i},X,i\right)$ is an arbitrary feature function over its arguments, and λ_{j} is a learnt weight for each feature function. Higher λ weights make their corresponding finite state machine transitions more likely.

Such as in HMMs, the parameters involved in CRF's have to be estimated, and a problem is thus to determine the parameters $\lambda=(\lambda_{1},\lambda_{2},...;\mu_{1},\mu_{2},...)$ from the training data $D=\left\{\left(X^{(i)},Y^{(i)}\right)\right\}_{i=1}^{N}$ with empirical distribution $\tilde{p}(X,Y)$, such that the log-likelihood of the training data is maximized. This is done using the maximum-likelihood method to estimate the parameters. This estimates can be efficiently derived using a first-order optimization algorithm such as the gradient descent method.

Finally, we also need to find the most likely sequence Y_{i} for the given observations X_{i}. Such as in HMMs, this problem can be solved using the Viterbi algorithm.

Some well known implementations of CRFs are provided in LingPipe, Mallet and MinorThird[1].

2.2 Evaluation Methods for Entity Recognition

The evaluation of information extraction systems is usually made by comparing the results of the automatic assignment done by systems, against the manual assignments done by human experts. Also, this evaluation can be done at the level of tokens or chunks. The first method states that the evaluation will be done token by token, i.e., each token is verified if it is, for example, a timex or not. The second method states that the evaluation will be done chunk by chunk, i.e., segments comprising of multiple tokens and the system verifies if, for example, that chunk forms an entire entity string, in our case a temporal expression. Typically used metrics are Accuracy, Precision, Recall and the F-measure. Let A be the true positives (i.e., the members attributed to a class by the system that really belong to that class) returned by the system, let B be the false positives (i.e. the members attributed to a class, by the system, that are not from that class), C the false negatives (i.e., the members that should had been attributed to a class but where the system failed in the attribution) and D be the true negatives (i.e., the members not attributed to a class by the system that do not belong to a class). The previous metrics can be defined as follows:

[1] http://sourceforge.net/apps/trac/minorthird/wiki

- **Accuracy** is as the proportion of correct assignments to a class in the set of all assignments, and is computed for every classes at the same time, while the metrics described next are computed class by class.

$$Accuracy = \frac{|A| + |D|}{|A + B + C + D|}$$

- **Precision** is defined as the proportion of members assigned to a class that really are from that class, and it is given by the formula:

$$Precision = \frac{|A|}{|A| + |B|}$$

- **Recall** is the proportion of class members that the system assigns to a class.

$$Recall = \frac{|A|}{|A| + |C|}$$

- The **F-Measure** makes a trade off between precision and recall, and corresponds to the harmonic mean of those two methods of evaluation. The F_1 performance measure penalizes systems that sacrifices one measure for another too drastically (Jurafsky & Martin, 2000).

$$F_1 = \frac{2 * (P * R)}{P + R}$$

2.3 Annotation Schemes for Temporal Expressions

Annotation schemes allow us to mark entities in textual documents. Along with the delimitation of the words, word chunks or sentences that belong to some class, we can also have several extensions that help to describe what has been marked.

In the context of this work, we are interested in annotation schemes for temporal expressions. Over the years several annotation schemes have been developed. One well known annotation scheme for temporal expressions is TIMEX2 (Wilson *et al.*, 2001), an evolution of the TIMEX scheme for annotating temporal references that was originally used in the 6th Message Understanding Conference. TIMEX2 defines an SGML element for annotating temporal references inline with the text, with six attributes that help gathering the semantics of a temporal expression. These attributes are as follows:

- VAL: This attribute contains a normalized value for the date, time or duration covered by annotated expression, in an format similar to that of ISO-8601. It can take one of three basic types of values, namely:
  - Specific points in time are expressed as a string matching the general pattern **ddd-dd-ddTdd:dd:dd.d+**, which can be interpreted as a sequence of **year-month-dateT hour:minute:seconds**. These strings may be truncated from the right, indicating points of coarser granularity. Any place in these strings can also be filled with a placeholder **X**, which indicates an unknown or vague value, and there are a handful of token values (i.e., character strings) for seasons and parts of the day which may substitute for months and times. There is also an alternate week-based format corresponding to the pattern **ddd-Wdd-d**, and which can be interpreted as **year-Wweek number-day of the week**.
  - Durations are expressed as a string matching the pattern **Pd+u or PTd+u**, where **d+** indicates one or more digits and **u** indicates a unit token (e.g., the character **Y** for years). A placeholder **X** may be used instead of a number to indicate vagueness.
  - Vague points in time are indicated as a string like PAST REF, PRESENT REF or FUTURE REF.

– MOD: This attribute captures temporal modifiers, using values such as BEFORE, AFTER, LESS THAN, MORE THAN, EQUAL OR LESS, START, MID, END or APPROX.

– ANCHOR VAL: This attribute contains a normalized value for an anchoring date or time, in a format similar to that of ISO-8601.

– ANCHOR DIR: This attribute captures the relative direction of orientation between the VAL and ANCHOR VAL attributes, as in WITHIN, STARTING, ENDING, AS OF, BEFORE or AFTER. It is used to express information about when a duration is placed.

– SET: This attribute identifies expressions denoting sets of times (i.e., recurrences), and it either takes the value of YES or is empty.

– COMMENT: This attribute contains any comments made by the annotators and it is ignored from the point of view of automated processing.

The TIMEX2 annotation scheme is used by most of the current systems dealing with the temporal expression resolution problem.

Another well known markup language for temporal and event expressions is TimeML (Pustejovsky *et al.*, 2003). It was first developed in the 2002 TERQAS (Time and Event Recognition

for Question Answering Systems) workshop, that set out to enhance natural language question answering systems to answer temporally-based questions about events and entities mentioned in news articles. TimeML aims to tag events, temporal expressions and the temporal relations between these events. It addresses four problems regarding event markup, including time stamping (through which an event is anchored to a time), ordering events with respect to one another, reasoning with contextually underspecified temporal expressions, and reasoning about the length of events and their outcomes.

To address these problems TimeML defines several tags, such as TIMEML, EVENT, TIMEX3, SIGNAL, etc. Each one of the elements has a different purpose, and together they gather as much information as possible about the temporal events, making this annotation scheme particularly powerful.

In my work, I will only address the recognition and normalization of temporal expressions, and thus I will use the TIMEX2 format instead of TimeML.

2.4 Summary

In this chapter, an overview of the fundamental concepts for information retrieval in general and temporal information resolution in particular is given.

As we can see from this chapter, machine learning methods are well suited for the problem of temporal information resolution, providing the basis for a portable and robust system.

We also addressed in this chapter, the evaluation methods more commonly used in the entity recognition area, as well as the annotation schemes more frequently used in the area of temporal information resolution.

Chapter 3

Related Work

Over the years, several researchers proposed different methods to deal with the recognition and disambiguation of temporal expressions in text. Some have chosen to use methods based on rules, while others used a mixture of rules with machine learning. All the proposed had their advantages and disadvantages. This section will describe some of the best known and most successful systems related to the resolution of temporal expressions in text.

3.1 The TempEx System

TempEx was the first TIMEX2 tagger (Mani & Wilson, 2000). It consists of a simple Perl tool that implements some heuristics based on part-of-speech tags (Abney, 1996), using finite state automata. The system also performs limited disambiguation on the recognized expressions (i.e., it only partially supports the TIMEX2 element).

TempEx was built with basis on a subset of the TIMEX2 annotation scheme, aiming to be simple enough to be understood and executed by humans. As such, several kinds of time expressions are not to be tagged and others, despite tagged, are not assigned a disambiguation value, because doing so would violate the simplicity of the system. For example, unanchored intervals, such as *half an hour* are not tagged. Non-specific time expressions like generics (e.g., *June* in *June is usually hot*) and indefinites (e.g., *a Monday*) are tagged without a value. Finally, ambiguous expressions without a strongly preferred reading are left without a value. The general modules that constitute the architecture of the system are described in Figure 3.3.

The system receives a document which has been previously tokenized into words and sentences.

It then sends the document into a first module, where the words are tagged with the corresponding part-of-speech tags, i.e., the words are tagged as corresponding to a particular part of speech, such as nouns, verbs, adjectives, etc. The system then takes the document to another module that resolves self-contained time expressions, i.e., time expressions that are fully specified, such as *June 2010* which is assigned a value of 21:10:06, meaning 21st century, year of 2010 and 6th month. These two modules use a variety of hand-crafted and machine-learnt rules.

Finally, the system passes the document to a module which resolves context-dependent time expressions using an ordered sequence of rules. The last module, as the name implies, is responsible for resolving time expressions that depend on the context in which they appear. The time value that a given context-dependent expression is related to is called *Reference Time*, and this *Reference Time* can be assigned a value corresponding to either a *Temporal Focus* or the document date. The document date is the creation date of the document, and the *Temporal Focus* is the time currently being talked about in the text. The last module uses an ordered sequence of rules to handle the context-dependent expressions that cover a limited range of expressions. Lexical triggers like *today*, *yesterday*, and *tomorrow*, as well as words which indicate a positional offset, like *next month* or *last year*, are used as lexical markers to describe the direction and magnitude of the offset from the *Reference Time*. Expressions like *Wednesday* in the phrase *the action taken Wednesday*, or bare moth names like $June$, are resolved based on the tense of neighboring past or future verbs, if any.

A Java port of the TempEx Perl engine is available as part of the QALL-ME[1] open-source framework for question answering. In this work, this system was used as a baseline.

Indeed, TempEx has mostly been used as a baseline for measuring the performance of a simple system over new datasets, or as a baseline for comparing systems.

3.2 The Chronos System

Chronos is a more complex system developed in the context of the TERN 2004 evaluation campaign (Negri & Marseglia, 2004). Text processing in Chronos involves tokenization, statistical

[1] `http://qallme.sourceforge.net/`

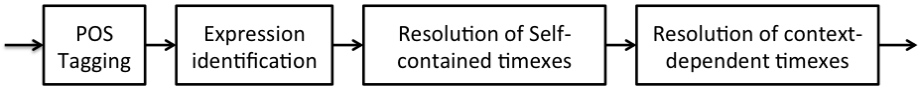

Figure 3.3: The Overall Architecture of the TempEx system

part-of-speech tagging, and multiword recognition. The system is designed to provide an automatic annotation of textual data with the TIMEX2 format. Figure 3.4 illustrates the overall architecture of the system Chronos.

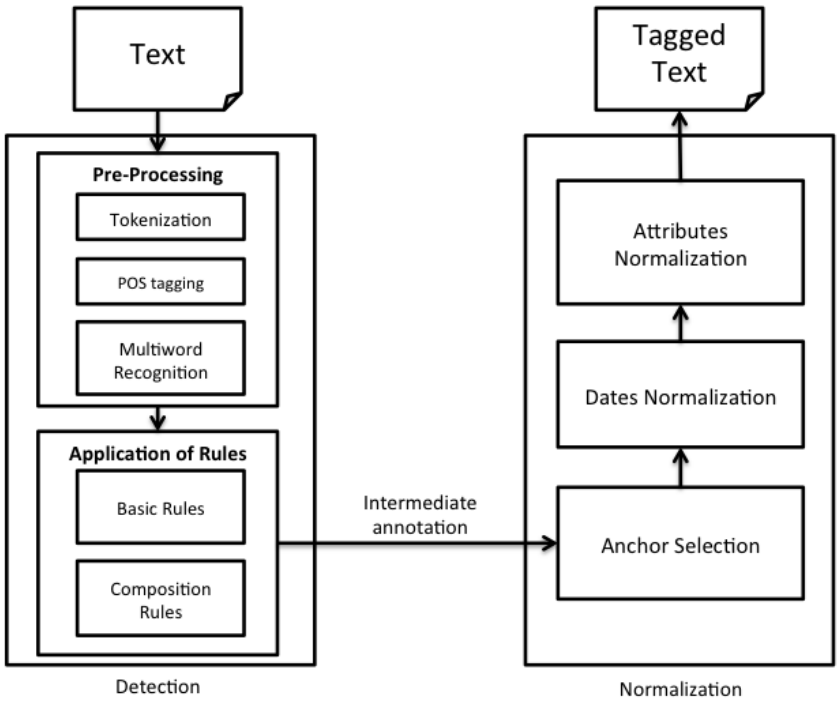

Figure 3.4: The Overall Architecture of the Chronos system

As we can see from Figure 3.4 the system relies on two main components, namely, Detection and Normalization. The detection component is in charge of the linguistic analysis of the input text, for the production of an intermediate annotation of that text. This component uses a rule-based approach, i.e., regular expressions, to detect the time expressions. The first step involves tokenization of the text and statistical part-of-speech tagging. Multiword recognition is also done in this phase, using a list of about five thousand multiwords terms (i.e., compounds, collocations and complex terms) automatically extracted from WordNet (C.Fellbaum, 1998).

The second step involves the application of basic and composition rules. The application of basic rules uses a set of approximately one thousand hand-crafted rules to detect and determine the extent of all possible time expressions in the text. This stage also gathers information about the expressions, which is latter used in the process of normalization. The application of composition rules consists of using a set of high-level rules which resolve ambiguities when multiple annotations are possible. Conflicts may occur when a recognized time expression overlaps with some other, or when it is adjacent to one or more other detected time expressions. For instance,

consider the sentence, *That man walked **the whole Monday night***. The basic rules recognize the following three time expressions: *the whole Monday*, *Monday night*, and *the whole Monday night*. The application of composition rules considers the starting and ending positions of the expressions to deal with these problems. Simply put, the system sees the start and end positions of the three recognized expressions and chooses the one with the largest extent (in this case, *the whole Monday night*). The first component of the system ends by returning an intermediate annotation of the input text using the TIMEX2 tag assigned to the temporal expressions.

At this time of the process, not all the produced information is in the form of the TIMEX2 attribute/value pairs. Additional attributes, namely *TYPE*, *T-CAT*, *QUANT* and *OP*, are assigned by the system for posterior use in the normalization phase. These attributes are used because, often, the superficial form of a time expression does not provide enough information for a correct normalization. These attributes will only appear in the intermediate annotation between the recognition and normalization phases.

The TYPE attribute is filled with one of the two possible values, namely *T-ABS* or *T-REL*. The first value is used to indicate absolute time expressions (e.g., *June 2, 2010*). The second value is used to indicate relative time expressions (e.g., *two weeks ago*) that require additional information to be normalized.

The T-CAT attribute is used only for relative time expressions, and indicates the granularity of the time expression. The possible values to be assigned to this attribute are *second, minute, hour, day, week, month, year, decade, century or millennium*. For example, for the time expression *two weeks ago* the value assigned to the *T-CAT* attribute would be *week*.

The QUANT attribute is used only for relative time expressions, and reflects the quantity that has to be added or subtracted for the calculation of final VAL attribute. This attribute takes an integer value $n >= 0$, reflecting the quantity. For example, for the time expression *two weeks ago* the attribute would be filled with the value 2.

Finally, attribute OP is also only used for relative expressions, reflecting the operation applied for the final calculation of the VAL attribute. This attribute can be assigned the values $+$,$-$, or $=$. For instance, for the time expression used in the previous examples, the value assigned to the OP attribute would be $-$.

The normalization component takes in the intermediate annotations and determines the correct values for each attribute of the detected time expressions, producing a tagged text compliant with the TIMEX2 annotation scheme. As we can see from Figure 3.4 this process is developed along three steps, namely anchor selection, date normalization and attribute normalization.

The date normalization step consists of assigning values to the VAL attribute of each detected

time expression. In this phase, the system sees, for each time expression, what value is assigned to the TYPE attribute. If the value is *T-ABS*, then the system, uses regular expressions to translate the value to the correct normalized form. For example, the time expression *7 June 2010* is transcribed into *2010-06-07* and then used to fill the VAL attribute. However, if the TYPE attribute is assigned with the value *T-REL*, which means that the time expression is relative, then the system requires additional information to fill the VAL attribute. This is where the other additional attributes come to play, respectively the OP and the QUANT attributes. These attributes, together with the value of the anchor, allow the correct normalization of the value. For example, given the relative time expression *three years later* and the anchor of that expression being *2010*, then the OP attribute would be filled with + and the *QUANT* with 3, leading the system to correctly fill the VAL attribute with the value of *2013*.

Anchor selection is a fundamental step in the correct resolution of relative time expressions. This phase consists in the connection of each detected relative time expressions (e.g., *three years later*) to an absolute value (e.g., *2010*) called the *anchor*. This process starts at the beginning of the document and then, for every relative time expression, tries to determine the corresponding anchors. As most of the relative time expressions have their anchors mentioned earlier in the document, in practice each relative time expression may represent the anchor for the following expressions given in the text. Anchor selection in Chronos follows two main strategies, namely *CR-DATE* or *PR-DATE*. The *CR-DATE* strategy associates to a relative time expression the document's creation date, normally found in the beginning of the document. The *PR-DATE* strategy associates, to a relative time expression, the value of the nearest previous absolute time expression with equal or higher granularity. This is where the attribute T-CAT comes into action. For instance, if we have a relative time expression like *three years later* and the last absolute time expression before that is *2010*, then the anchor for the relative time expression would be *2010* because it has the same granularity.

When these attributes are not explicitly specified, as it happens in the phrase *He started training on June 5 2007, and he raced **the following Friday***, then additional effort has to be undertaken. The system has to determine what day of the week corresponds to the time expression *June 5 2007*, which is Tuesday, latter applying the operator + because of the word *following*. The system adds three days, which are the number of days separating the anchor from the expression *the following Friday* and, in the end, the system computes the final date associated to the relative time expression, filling its VAL attribute with the correct value of *2007-06-08*.

Finally, the attribute normalization step produces the tagged text, removing the temporary attributes introduced in the recognition phase.

The Chronos system achieves high precision, due to its rule-based approach, although recall

leaves much to desire, due to the inherent difficulty in covering all the possible ways to express time expressions. Also, the approach taken to deal with conflicts was the source of many errors. Despite the previous problems, we have that Chronos system reflects the benefits and disadvantages of using a carefully tuned hand-crafted rule approach.

3.3 The TERSEO System

The TERSEO system was developed at the Natural Language Processing and Information System Group at the University of Alicante in 2006 (Saquete *et al.*, 2005). This system aims to order events based on temporal information resolution. The system processes documents in two main steps. First, it performs the recognition and resolution of temporal expressions. Second, based on the resolved time expressions, it establishes an order between the events that contain them (Saquete *et al.*, 2005). The first step is one that most concerns this work. Figure 3.5 shows the overall architecture of the TERSEO system.

As we can see from Figure 3.5 the system begins by passing the document to the temporal information detection unit, which is divided into two main tasks. The first task is temporal expression recognition, and the second task is signal recognition. These two tasks are now described.

Although both steps are independent, they share a common preprocessing of the text. Before the document is passed to each of the steps, it is tagged with lexical and morphological information by a POS Tagger. The tagged text is then input to a temporal parser. This parser uses a grammar based on two different kinds of rules. The first rules cover the recognition of explicit dates and times (e.g., *18/11/2010*), while the second rules cover implicit temporal reference recognition (e.g., *tomorrow*). This grammar recognizes a large set of different date and time formats (Saquete *et al.*, 2003).

After this tagging, the system starts with the detection of temporal expressions and temporal signals. The first task, detection of temporal expressions, aims to recognize time expressions according to two temporal expression classifications proposed specifically for TERSEO.

The first classification is based on the type of reference, and is divided into two types, namely explicit temporal expressions and implicit temporal expressions. Explicit temporal expressions can be complete dates with or without time expressions (e.g., *18/11/2010, November 18th, 2010*), dates of events, noun phrases with explicit dates (e.g., *2010/2011 course*), or noun phrases with a well-known date (e.g., *Christmas*). Implicit temporal expressions can be expressions that refer to the document date, such as adverbs or adverbial phrases (e.g., *tomorrow*), noun phrases (e.g., *the next week*), prepositional phrases (e.g., *in the last month*), or expressions that refer to

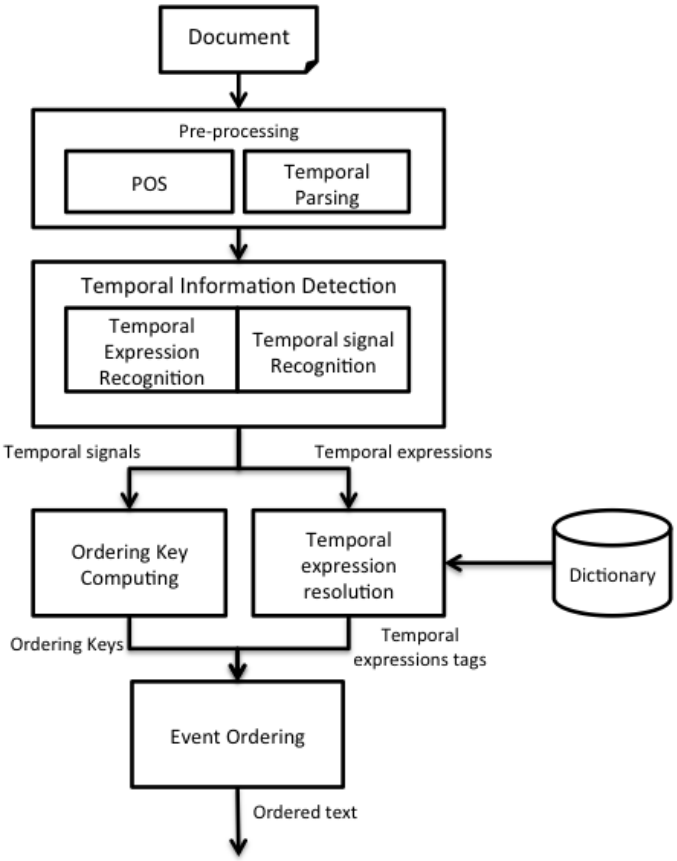

Figure 3.5: The Overall Architecture of the TERSEO system

another date, such as *during the course* or *after the next Christmas*.

The second classification is based on the representation of the temporal value of the expression. The temporal value can be *Concrete*, i.e., giving back a concrete day and/or time with a format corresponding to *mm/dd/yyyy (hh:mm:ss)* (e.g., *tomorrow*), *Period*, giving back a time interval or range of dates corresponding to [*mm/dd/yyyy-mm/dd/yyyy*] (e.g., *during the following five days*), or *Fuzzy*, giving back an approximate time interval because the system does not know the concrete date that the temporal expression refers to. This latter type is further divided into two types, namely *Fuzzy concrete* if the given result is an interval but the time expression refers to a concrete day within that interval and we do not know it for certain (e.g., *one last week*), and *Fuzzy period*, if the time expression refers to an interval contained within the given interval (e.g., *some days before*).

The second task, temporal signal detection, aims to detect temporal signals in the document. These signals relate the different events in the document and establish a chronological order

between these events. Some of these signals are *when, after, during, while and since*.

After these two tasks have finished, the system passes the document to the temporal expression resolution unit and to the ordering key computation unit, which are executed simultaneously.

The temporal expression resolution unit is divided into two different tasks, namely anaphoric relation resolution based on a temporal model, and tagging of temporal expressions.

The anaphoric resolution of the temporal expressions uses an inference engine that interprets every reference named earlier in the document. This unit accesses to the right entry in the inference engine in each case, and then applies the specified function obtaining a date in the format *dd/mm/yyyy* or a range of dates. These references can be estimated using the document date, here called (*FechaP*) or estimated using a date named before in the document being analyzed, called (*FechaA*). In this last case, a temporal model is used to determine on what date the dictionary operations will be done. This model is based on two essential rules. The first rules states that, by default, the date provided on document metadata (e.g., a publication date) is used as a base referent, if it exists. If not, the system date is used. The second rule states that if a non-anaphoric temporal expression is found, it is stored as a *FechaA*. This value is updated every time a non-anaphoric temporal expression appears in the text. Table 3.1 shows two examples of the entries that can be found in the dictionary.

Table 3.1: Examples of entries in the dictionary

REFERENCE	DICTIONARY ENTRY
'yesterday'	Day(FechaP)-1/Month(FechaP)/Year(FechaP)
'some days later'	>>>>FechaA

The tagging task assigns a tag to every temporal expression in the document. The tagging scheme follows a set of XML tags developed specifically for TERSEO, but it can be easily transformed to other existing schemes, like *TimeML*. The tagging scheme defines two tags, namely *DATE_TIME* for explicit temporal expressions, and *DATE_TIME_REF* for implicit time expressions. Both tags have a numeric attribute *ID* that identifies the expression. The attribute *VALDATE1*, *VALDATE2*, *VALTIME1* and *VALTIME2* store the range of dates and times obtained from the inference engine. The attributes *VALDATE2* and *VALTIME2* are only used to specify ranges, and *VALTIME1* can be omitted if only a date is specified. The last attribute is *VALORDER*, which is filled with the ordering value to be used in the event ordering unit. Also, only the *ID*, *VALDATE1* and *VALORDER* attributes are required.

In the ordering keys computing unit, the temporal signals obtained by the temporal signal detection unit are used to obtain the ordering keys. Each one of the temporal signals denotes a

relationship between the dates of the event that the signal is relating. For example, if we imagine that we have one date related to one event and another date related to a second event the signal between them will establish a certain order between these two events. This order will be established by the event ordering unit. Table 3.2 shows example ordering keys.

Finally, the event ordering unit, using the XML tags and the ordering keys produced by the earlier units, builds a table specifying the the order and the dates, if any, of every event. The order of every event is established according to two rules. The first rule corresponds to *event1* is previous to *event2* and is applied if the range of *VALDATE1*, *VALTIME1*, *VALDATE2*, *VALTIME2* associated with *event1* is prior to and does not overlap with the range associated with *event2*, or if the ordering key that relates both events is *event1*$<$*event2*. The second rule states that *event1* is concurrent to *event2*, and is applied if the range of *VALDATE1*, *VALTIME1*, *VALDATE2*, *VALTIME2* associated with *event1* overlaps the range associated with *event2*, or if the ordering key that relates both events is *event1*$=$*event2* or *event1i*$<=$*event2*$<=$*event1f*, *i* being initial and *f* final. The system assigns a sequential number to all events in the table, and assigns the same order number for concurrent events.

TERSEO is a system were the resolution of temporal information is not the ultimate purpose. Instead, the authors use that resolution to order temporal events in the text. Notwithstanding this, the system uses an interesting mixture of grammars, rules and dictionaries with an efficient tagging scheme. TERSEO is available online for anyone who wants to use it[1].

Table 3.2: Output of the ordering key obtaining unit example

SIGNAL	ORDERING KEY
After	DATE1 $>$ DATE2
When	DATE1 = DATE2
Before	DATE1$<$ DATE2
During	DATE2i $<=$ DATE1 $<>$ DATE2f

3.4 The DANTE System

DANTE[2] is another example of a rule-based system that participated in TERN (Mazur & Dale, 2007). As we can see in Figure 3.6 the system has two separate modules, namely a recognition module and interpretation module. The recognition module uses GATE's JAPE grammar

[1] http://gplsi.dlsi.ua.es/~stela/TERSEO/
[2] http://platypus.ics.mq.edu.au:8180/~mpawel/demos/dante/dante.jsp

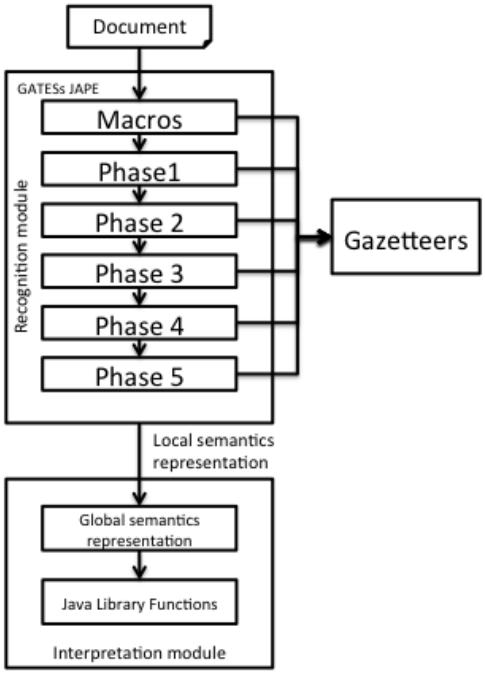

Figure 3.6: Overall Architecture for the DANTE system

formalism (Cunningham *et al.*, 2002), and consists of five executed phrases, over a document in sequence. The first phase consists of 80 macros used in grammar rules that are textually copied into bodies of rules and compiled to Java code. The other five phases contain rules which match annotations introduced by earlier components (for example, the tokenizer or POS tagger). A total of 31 gazetteers with a total of 1418 entries with names of days, months, time zones, and so on, are also used. In the end of the recognition module, the extents of the temporal expression and their local semantics are outputted. Local semantics refer to a level of representation that corresponds to the semantic content that is derivable directly from the text representation of temporal expressions.

Finally the system enters the interpretation module, where it will normalize each temporal expression. This process steps through a document sentence by sentence, and begins by transforming the local semantic representation, resulting from the previous module, into a global semantic representation expressed by means of TIMEX2 annotations. This module uses a library of Java functions for various calculations on dates and times, namely to unify the temporal expression with some reference date and to add or subtract a specified number of units to or from a reference date, resulting in the normalization of each temporal expression.

DANTE has been evaluated in the ACE 2007 TERN task with promising results, see Table 3.5.

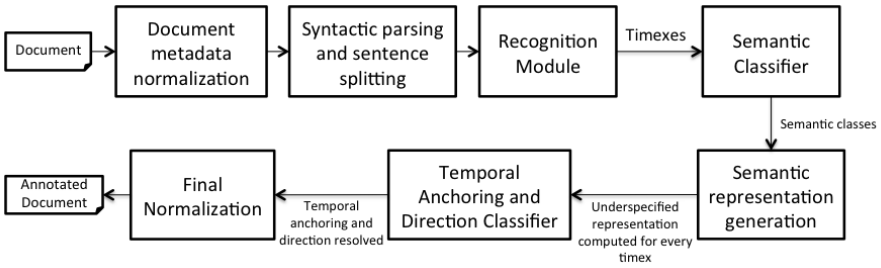

Figure 3.7: The Overall Architecture of the TimexTag system

3.5 The TimexTag System

The TimexTag system, unlike the other systems described earlier, minimizes the use of hand-crafted rules by using machine learning methods in both the recognition and normalization tasks (Ahn *et al.*, 2007). Like most of the systems described earlier, this system makes use of the TIMEX2 annotation scheme for temporal expression annotation. Figure 3.7 depicts the overall system architecture for TimexTag.

As we can see from the figure, the input data passes through several modules before the system produces an annotated document. The system begins with the recognition and normalization of timexes in the document metadata. This is done using a set of regular expressions, 14 in total, to extract the document timestamp to be used in the following modules. Then, the system parses the document, i.e. the document is divided in phrases and each phrase of the parsed document goes through the first module, the recognition module. In this module, the recognition task is reduced to a binary phrase classification, in which syntactic constituents are classified as timexes or non-timexes. For this classification, the TimexTag system uses a support vector machine method, in particular the LIBSVM[1] linear kernel implementation (Chang & Lin, 2001). Also, the lexical categories of the elements in the text are parsed using the Charniak (2000) parser. These lexical categories are used to identify candidate phrases and to extract parse-based features. Some examples of the features that are extracted are character type patterns, lexical features such as weekday name and numeric years, a context window of two words to the left, phrase type, the phrase head and initial word, POS tags, and the dependency parent of the head.

In the semantic classifier module, semantic classes are given to each timex recognized by the earlier module, namely the class *recur* for recurrences, *gendur* for generic or vague durations,

[1] `http://www.csie.ntu.edu.tw/~cjlin/libsvm/`

duration for durations, *genpoint* for generic or vague point timexes, *point* for point timexes, or *null* for other cases. As semantic classes are not annotated by the TIMEX2 standard, we have that the semantic classes used for normalization are inferred from the values of the attributes that are annotated, namely the attributes VAL and SET from TIMEX2. The semantic classes are assigned using the attributes annotated as follows:

- the *recur* class is attributed if the SET attribute is set to true.
- the *gendur* class is attributed if the VAL attribute begins with PX or PTX.
- the *duration* class is attributed if the VAL attribute begins with P[0-9] or PT[0-9].
- the *genpoint* class is attributed if the VAL attribute begins with T[0-9], if VAL is one of $past\_ref$, $present\_ref$ or $future\_ref$, or if the VAL attribute has an unspecified high-order position (i.e., if the millennium position is X).
- the *point* class is attributed if the VAL attribute has a specified high-order position, although it may be precise or not, i.e., may include X at other positions.
- the *null* class is attributed if the VAL attribute is not present in the timex.

This classification, is made through an SVM model with a linear kernel, just like in the recognition module.

In the next iteration of the system, the semantic representation generation module comes into action. This module uses regular expression patterns to compute, for each timex recognized earlier, a typed feature structure that depends on the timex's semantic class. Table 3.3 shows, for each semantic class, an example of a rule used by the system.

Next, the temporal anchoring and direction classification module is used to determine, for each point and genpoint timex, its temporal anchor and the direction class between them. To recognize the temporal anchor of a timex, the system uses a simple heuristic method, in which, for deictic

Table 3.3: Examples of rules used by the semantic representation generation module

class	example
dur	`Numeric-? (UNIT | UNITS)`
gendur	`(UNIT | UNITS)`
genpt	`(NUM24 | NUMWORD) o'clock`
point	`^ Approx? DAYNAME? MONTHNAME.? Num31OrRank ,? YearNum`
recur	`(every | per) Numeric UNITS`
misc	`NUMWORD ((and | -)? NUMWORD)*`

timexes, such as *today* or *three years ago*, the document timestamp is used, and for some anaphoric timexes, such as *two months earlier* or *the next week*, the most recent point timex, if it is fine-grained enough, is used as the temporal anchor. Otherwise the document timestamp is used. Since the corpora over which this system was initially used were composed of short news texts, the system actually treats anaphoric name-like points (i.e., timexes providing a position in a cycle, such as, a day name) as deictic timexes and the most recent timex is only used for anaphoric offsets, i.e. *timexes* such as *two months earlier*.

The direction classification, between a timex and its anchor, is computed using the VAL attribute. After determining the anchor of a point or genpoint class timex, the VAL attribute of the timex and its anchor are compared to determine the direction class of the timex. Again an SVM model with a linear kernel is used for this classification, but this time two sets of features are added to those used in the recognition and semantic classification. The first of these sets of features uses the verb tense of the closest verb and its POS tags, and others verbs directly related to this verb. The second set compares day names, month names, and years to the document timestamp. In the end of this procedure, the classes attributed by the classification are *after*, *before* and *same*, respectively if the timex refers to a time that succeeds, precedes or is the same as its anchor.

Finally, the last module, combines the semantic representation, temporal anchoring, and direction class to compute the final normalization.

TimexTag, aimed to minimize the use of hand-crafted rules, by means of machine-learned methods. The overall performance of the system stands side by side with that of state-of-the-art systems and, surprisingly, the system performance is limited not by the normalization part but by the recognition part. The decoupled recognition and normalization tasks allow the system to use machine-learned methods on part of the problem, thus decreasing the use of hand-crafted rules.

3.6 The Temporal Expression Anchorer System

Unlike the systems described earlier, that try to normalize time expressions in newswire texts, the Temporal Expression Anchorer system (TEA), aims to normalize time expressions in emails (Han & Levin, 2006). This novel application is very pertinent, since there is an increasing demand for more sophisticated NLP applications capable of understanding email texts.

In this work, the authors also make very interesting comparisons between newswire text and emails. The corpora used for this work was the CSpace email corpus (Kraut *et al.*, 2005), which contains approximately 15,000 email messages collected from a management course conducted

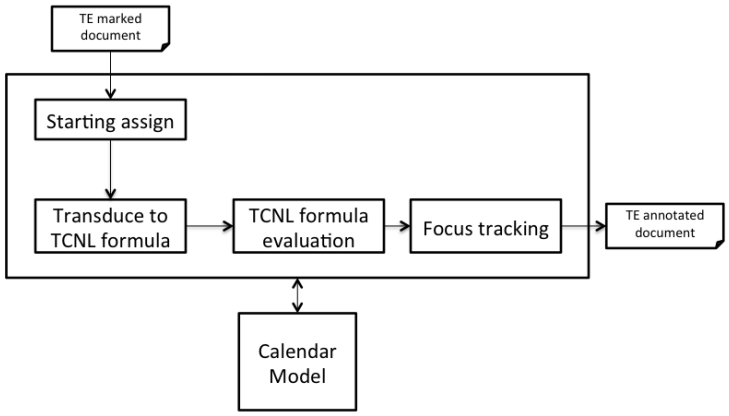

Figure 3.8: The TEA Overall Architecture

at Carnegie Mellon University in 1997.

Emails present several challenges that newswire text does not present. The first big difference is that the percentage of explicit expressions (i.e., those that can be directly normalized) occurring in emails are much lower than in newswire texts, namely about 25% for newswire texts and 9.5% for the email data sets considered. This is mainly due to economic reasons when writing emails. Another interesting difference encountered by the authors is that as emails are a communication medium, and thus they can be used as a reply attached within another email, or even used to address multiple recipients. This complicates the task of normalizing time expressions in emails. Also, ambiguities tend to appear more often in emails (e.g., *I'll be home at 2.*) and people tend to be more creative when writing emails, using abbreviations, lists, different month/day formats (e.g., *1/9* can mean *January 9* or *September 1*). Emails also contain more human errors such as misspellings (*"Thursday"* to mean *"Thursday"*). All those particularities, make the task of temporal expression resolution in emails very difficult.

The Temporal Expression Anchorer system uses a representation called *Time Calculus for Natural Language (TCNL)*, which presents a constraint-based representation of time, different from the representation encountered for instance in TIMEX2 (Han, 2003). Figure 3.8 shows the overall architecture of the TEA system.

The TEA system only does normalization of temporal expressions, and it begins by receiving an English text with temporal expressions already marked up using rules developed with the MinorThird[1] text mining package, and then corrected manually by two of the authors. Then, in the first module, the *speech time*, normally associated with deitic expressions like *tomorrow*, and the *temporal focus*, normally associated with relative expressions like *the next day*, are first assigned

[1] http://sourceforge.net/apps/trac/minorthird/wiki

to a timestamp, normally the received date of an email. Then, for each temporal expression and using the POS tags of the sentence, the nearest verb chunk is identified and if it is a past or present imperfective tense, the *"-p"* prefix will be assigned to its TCNL formula, otherwise the *"+f"* prefix is assigned.

A finite-state parser is then used to transduce each temporal expression into its TCNL formula. Table 3.4 shows some examples of TCNL formulas.

Expression	TCNL formula
Thursday night	+f{thu,night})
by Saturday Noon	[f +f{sat,noon}]
until AFTER midnight	[f{>=-p{midnight}}]

Table 3.4: Example of formulas for TCNL

After the TCNL formulas have been produced, the system evaluates them with the speech time and the current focus. Here the ambiguities are treated with heuristics:

1. Any candidate resulting in an inconsistency when computing a solution is removed.
2. If the result is meant to be a coordinate, i.e., a set of assignments to temporal units, such as *Friday the 13th*, than the one that is closest to the focus is chosen.
3. If the result is supposed to be an enumeration, i.e., a set of time points, like in *Friday and Wednesday*, the one whose starting point is closest to the focus and whose length is the shortest one, is chosen.
4. If none of the previous rules apply, the first candidate is chosen as the result.

Finally, the focus tracking module is used to determine if the result obtained in the previous module can replace the current focus. This is done using heuristics, namely, in cases where the result is an ambiguous coordinate (i.e., it denotes a possible range of points), if one of the bounds is *min* or *max*, (i.e., the minimal and the maximal time points TEA can reason with) than we use the other bound to be the new focus. If that is not possible the focus remains unchanged. On the other hand, if the result is an enumeration, a similar procedure is used to avoid using an enumeration with min/max bound as the new focus. Finally, no quantity can become a focus.

At all stages, the Calendar Model provides necessary services such as determining the granularity of expressions, comparing two expressions chronologically, and solving constraint satisfaction problems induced by TCNL formulae. This calendar is viewed as a constraint resolution system, where temporal units are treated as variables with a discrete, finite and fully ordered domain (e.g.,

the domain of unit *month* consists of *jan* $<...<$ *dec*). A temporal expression in natural language in effect assigns values to some of these units. The technical report by Han *et al.* (2006) provides additional details on TCNL calculus.

The TEA system presents a novel approach, by trying to deal with the normalization of time expressions in emails, unlike most of the other systems in the area which deal with newswire texts. This system also demonstrates that the community is trying to cope with the resolution of time expressions following many different approaches.

3.7 Reference Time Dynamic-Choosing

The work by Zhao *et al.* (2010) focus only on the normalization problem. In this work, the authors normalize temporal expressions using a time choosing mechanism called *reference time dynamic-choosing*, which assigns the appropriate reference times to different classes of implicit temporal expressions dynamically when normalizing.

The authors define that the implicit time consists on a modifier and the temporal noun modified by that modifier, although the modifier can be null. As shown in Figure 3.9, the temporal reference comes from two parts, namely a modifier reference and a temporal noun reference. As the former is inferred from the latter, the temporal noun reference plays a more important role in normalization. In practice, the system takes the report time or the nearest narrative time in the text as the reference time of the temporal noun, when normalizing a whole implicit time. In consequence, the system assigns one of two distinct classes to the implicit time reference, corresponding to the temporal noun class, namely Global Temporal Noun, which takes the report time or creation time of the document as the reference time, and Local Temporal Noun, which makes reference to the nearest narrative time in the text.

For the reference time choosing, and unlike other systems that use static rules to find the reference time, or even the document creation time for all expressions, this system maintains a table to hold full reference times. This table is updated and maintained dynamically after each normalization process. The table consists of two parts, namely Global Reference Time and Local Reference Time. Figure 3.10 exemplifies the usage of this table, where one can see that the system updates the table each time a temporal expression is normalized, and the temporal reference used to normalize the expression is either a local reference time, if the time expression is to be normalized according to the narrative time, or a global reference if the document creation time is to be used to normalize the expression.

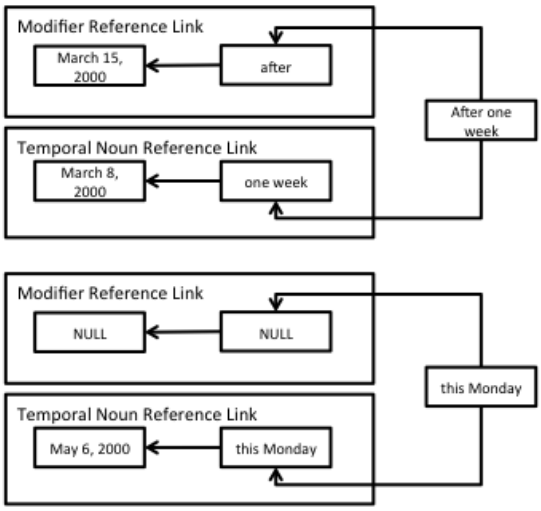

Figure 3.9: Usage of reference table

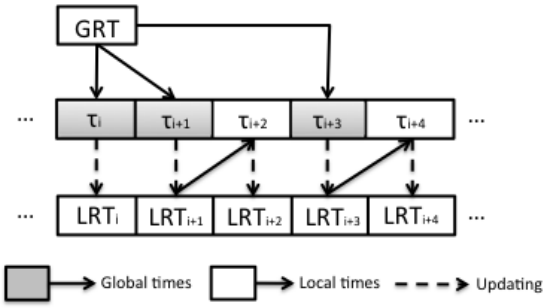

Figure 3.10: Example reference link

The basic normalization algorithm of this system works as follows: first, the document creation time is used to initialize both the Global Reference Time and the Local Reference Time. Then, for each temporal expression in the text, a segmentation is performed for dividing the temporal expression into modifier and temporal noun. If the temporal expression is explicit, then the Local Reference table is updated and the temporal expression is directly normalized. If the temporal expression has a local noun, then the latest value from the Local Reference table is retrieved and the value is then normalized. Finally, if the temporal expression has a global noun, then the latest value from the Global Reference table is retrieved, the Local Reference table is updated, and the temporal expression is normalized.

Before this normalization process, the system passes each temporal expression through a defuzzification process, which aims to find the granularity and offset of each temporal expression. This process works as follows: for each temporal expression in the text, the system tries to get

its granularity. If this is not possible, it means the temporal expression is fuzzy. If the temporal expression is fuzzy then the system assigns a granularity corresponding to the granularity of the former temporal expression in the text. If the temporal expression is the first in the paragraph, then the system has to consider two cases checkings, if the temporal expression is the first in the first paragraph of the document, or if the temporal expression is the first of a non first paragraph. In the latter case the system assigns the coarser granularity between the last temporal expression in the previous paragraph and the next temporal expression. In the former case, a default granularity is retrieved from a dictionary. Finally, the system finds the offset of each temporal expression in a dictionary.

After finishing the defuzzification for the whole text, the basic normalization process is evoked. This system was run in a Chinese news corpus leading to promising results see Table 3.5.

3.8 Bootstrapping on Temporal Expressions

Poveda *et al.* (2009) described an experimental system that only deals with the recognition of temporal expressions and uses a semi-supervised (bootstrapping) approach to the extraction of temporal expressions. As we can see from Figure 3.11, the bootstrapping algorithm passes through several stages, each of them described next. The inputs to the bootstrapping algorithm are the unlabeled training corpus and a file of seed examples. The unlabeled corpus is a large collection of documents which has been tokenized, POS tagged, lemmatized, and syntactically analyzed for finding basic syntactic constituents (shallow parsing) and headwords. The second input is a set of seed examples, consisting of a series of token sequences which we assume to be correct time expressions. The seeds are supplied without additional features and without context information.

The bootstrapping algorithm works with two views of the same target data, namely *patterns* and *examples*. A *pattern* is a generalized representation that can match any sequence of tokens meeting the conditions expressed in the pattern. An *example* is an actual occurrence of a time expression. Patterns are generated from examples found in the corpus and, in turn, new examples are found by searching for matches of new patterns. Both patterns and examples may carry contextual information (i.e., a window of tokens to the left and right of the candidate time expression).

Initially, a single pass through the corpus is performed in order to find occurrences of the seeds

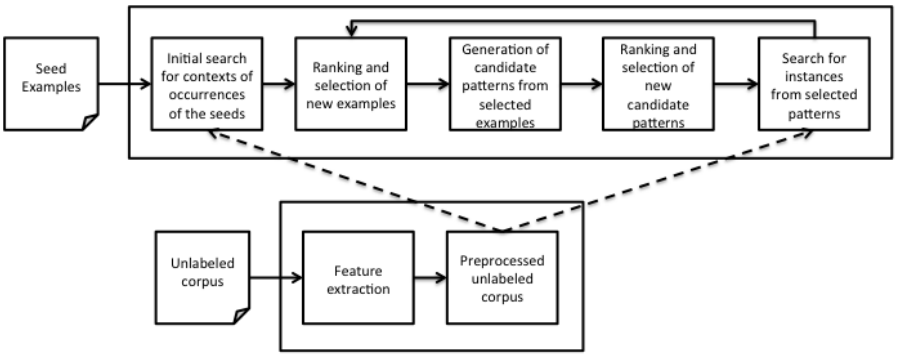

Figure 3.11: Bootstrapping algorithm architecture

in the text (i.e., the system bootstraps an initial set of *examples*). After this first step, the bootstrapping process consists of a succession of iterations as follows.

1. Ranking and selection of examples: In this step, each example produced during any of the previous iterations, 0 to $i-1$, is assigned a score. The top n examples are selected to expand the set of output examples and will be used for the next step.

2. Generation of candidate patterns: In this step, candidate patterns for the current iteration are generated from the selected examples of the previous step.

3. Ranking and selection of candidate patterns: Here, each pattern from the current iteration is assigned a score and the top m patterns are selected to grow the set of output patterns and to be used in the next step. Also in this step involves a process of analysis of subsumptions, performed simultaneously with selection, in which the set of selected patterns is examined and those that are subsumed by other patterns are discarded.

4. Search for instances of the selected patterns: In this step, the system traverses the training corpus, in order to search for instances (matches) of the selected patterns, which, together with the accepted examples from all previous iterations will form the set of candidate examples for the next iteration.

For the evaluation of this system, the authors used the NW (newswire) category of LDC's ACE 2005 Unsupervised Data Pool, and simultaneously the ACE 2005 labelled corpus. While this last corpus was split in two halves, one half was used obtain the initial seed examples, while the other was used for evaluation. The results are presented in Table 3.5.

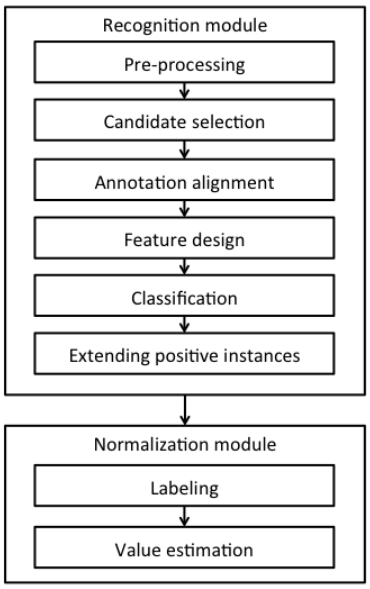

Figure 3.12: KUL Overall architecture

3.9 The KUL System

The Katholieke Universiteit Leuven (KUL) Java-based system was developed in the context of a participation in the Temp-Eval-2 event (Kolomiyets & Moens, 2010). As we can see in Figure 3.12, the system follows a pipelined method for information processing that can be split into two main modules, namely recognition and normalization.

In the recognition module, the pre-processing sub-module submits the input text modules to syntactic analysis, performing sentence detection, tokenization, part-of-speech tagging and parsing. In this sub-module, the OpenNLP[1] package is used.

The candidate selection sub-module filters chunk-phrases that cannot belong to temporal expressions according to their lexical categories. In this module, only chunk-phrases belonging to specific categories (i.e., nouns, proper names, noun phrases, adjectives, adjective phrases, adverbs, adverbial phrases, and numbers) are passed to the next stage.

In the annotation alignment sub-module, and if the system is used for training classifiers, all the candidates in a sentence are examined against the available annotations. The candidates whose parse and annotation extents align are taken as positive examples, and the remaining are considered as negative.

[1] `http://opennlp.sourceforge.net`

The feature design sub-module, as the name implies, produces boolean features extracted from the text. The features considered in this module are as follows:

- Last token in the phrase, i.e. the most probable token to be a temporal trigger.
- Lemma of the last phrasal token.
- Part-of-speech tag for the last phrasal token.
- Character pattern of the last phrasal token.
- The part-of-speech tags for the last phrasal token and for its preceding token.
- Character pattern of the entire phrase.
- A concatenated string of sub-parse types for the phrase.
- A Boolean feature indicating nested complex phrasal parses, such as noun verb, adverbial, adjective or prepositional phrase.
- The number of the nested sub-parses to the deepest preterminal sub-parse.

In the classification module, a maximum entropy classifier is used to classify each temporal expression in the text, according to their probability of belonging to a certain class.

Finally, in the recognition module, and to deal with the sparseness of annotated corpora, words that do not occur in the training set are associated to words that do occur in the training set according to their similarity. In order to find these words, the authors used a latent word language model (LWLM) (Deschacht & Moens, 2009), which is based on a Hidden Markov Model approach for estimating the latent word parameters. WordNet is also used as a source of information for obtaining a complete set of words similar to the given one.

The normalization module uses a rule-based approach and begins in the labelling sub-module, which provides tags to tokens from a defined set of tags. Several categories, based on the semantics of temporally relevant information and based on simple syntax were defined, namely ordinal numbers (*first,* 30^{th}*, etc.*), cardinal numbers (*one, 10, etc.*), month names (*Jan., January, etc.*), week day names (*Monday, etc.*), and so on. For each category, a vocabulary is constructed, in which each entry specifies a value of a temporal field, a final date/time value, or a method with parameters to apply.

During labelling, each token in a temporal expression is tagged with one of the multiple labels corresponding to the categories defined above. For each category, a custom detector declares the method to run and the expected type of the result. As output, this sub-module provides labels

of the categories to the tokens in the temporal expression. If there is no entry in the vocabulary for a token, its part-of-speech tag is used.

The final sub-module of the system is called value estimation. This sub-module aggregates the values defined for entries in the vocabulary, and/or executes instructions or methods specified. Also, a set of predefined resolution rules is provided, and thus can be extended with new implementations of resolution strategies. To resolve complex relative temporal expressions, additional information, from the recognition step is used. This information includes semantic types for the *timexes*, discourse type, and contextual temporal information.

The evaluation results over the TempEval-2 challenge are presented in Table 3.5.

3.10 The HeidelTime System

HeidelTime is a rule-based system that achieved the best F1 score in the extraction and normalization of temporal expressions over the TempEval-2 challenge (Strötgen & Gertz, 2010).

HeidelTime is implemented as a UIMA[1] (Unstructured Information Management Architecture) component, that integrates an existing processing pipeline, as depicted in Figure 3.13. Note that some components are purposely left out of the picture, since we only want to analyze the recognition and normalization tasks.

As we can see from the Figure 3.13, the UIMA pipeline goes through three initial steps before entering the HeideTime component, namely sentence splitter, tokenizer and POS tagger. As the name implies, the first step splits the document into sentences, the second splits the sentences

[1] http://uima.apache.org

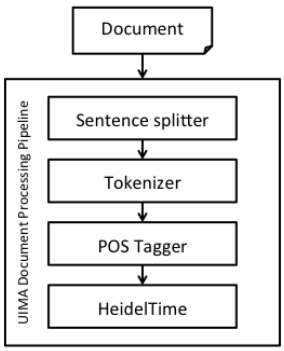

Figure 3.13: UIMA Overall architecture

into tokens, and the third component uses the OpenNLP part-of-speech tagger to assign the corresponding part-of-speech tag to each token. The information gathered from these three initial components is then used by the HeidelTime tagger to extract and normalize temporal expressions.

The HeidelTime system, sees each temporal expression as triple $t_{ei} = (e_i, t_i, v_i)$, where e_i is the expression itself, t_i represents the type of the expression, (i.e., *Date, Time, Duration and Set*), and v_i is the normalized value. Thus, the objective of the system is to extract fot each temporal expression the expression e_i, and to correctly assign the type attribute t_i and the value attribute v_i. For this purpose, the HeidelTime system uses hand-crafted rules, grouped into the four types mentioned earlier.

The extraction rules mainly consists of regular expressions patterns. However, other features can also be used, e.g. a constrain on what part of part-of-speech the next or previous token has to have. Also, HeidelTime contains resources for weekdays, months, or seasons, which are realized as regular expressions that can be accessed by multiple extraction rules. In addition, there are knowledge resources for the normalization of such expressions.

The normalization task in HeidelTime is done in two separate ways, depending if the temporal expression is explicit, implicit or relative. The first two types are computed almost directly, and the third type is computed in a post-processing step.

For explicit temporal expressions (i.e., *March 11, 1982*) rules are applied, directly converting that temporal expression to the normalized value.

For implicit temporal expressions, (i.e., *Independence Day 2010*), resources are used to extract the date associated with *Independence Day*, and afterwards rules are used to combine these with the year 2010, resulting in the normalized value of *2010-07-04*.

For relative temporal expressions, the system has to compute the reference time of the relative temporal expression. HeidelTime assigns the values in an underspecified format depending on the assumed reference time, and disambiguates them in a post-processing step. The underspecified values, for the examples, are UNDEF-last-June, UNDEF-June, and UNDEFREF-last-year, respectively. For the first two examples, the document creation time (DCT) is assumed to be the reference time while for the last example the previously mentioned date is used for reference.

The post-processing step has two tasks, namely normalize underspecified value attributes and remove all extracted timex annotations that are invalid.

In the first post-processing task if the value starts with UNDEF-REF, the previously mentioned date is used for disambiguation, otherwise the document creation time if meaningful. The value

UNDEF-last-June is disambiguated by calculating the June before the document creation time. More complex are even less underspecified values like UNDEF-June. Here, linguistic knowledge is used to disambiguate which June is meant: The tense of the sentence is determined by using the part-of-speech information of the tokens and checking the semantics of the verbs in the sentence. This method identifies whether a sentence is past, present, or future tense. For example, the tense of the sentence *In June, new results will be published* will be determined to be future tense and the new value UNDEF-next-June can be assigned instead of UNDEF-last-June if past tense was identified. Such values are then disambiguated using the methods described above.

The last post-processing step is to remove all extracted timex annotations that are invalid. Invalid are all expressions that are included in other expressions. For example, having the phrase *June 11* the whole phrase is found by a rule as well as just *June*. Since *June* is in *June 11*, it is removed.

3.11 Summary

From chapter 3 we can see several systems that reflect the efforts made by several researchers in order to improve temporal expressions resolution systems. Table 3.5 shows a comparison between all the systems surveyed here.

Table 3.5: Described Systems Comparisons

System	Recognition				Normalization				Corpus
	P	R	F_1	A	P	R	F_1	A	
TempEx	–	–	–	–	83.7%	82.7%	83.2%	–	NewsWire
Chronos	97.6%	88.0%	92.6%	–	87.5%	87.0%	87.2%	–	TERN2004
TERSEO	–	–	–	–	82.8%	85.5%	84.2%	–	NewsWire
DANTE	99.8%	98.0%	98.9%	–	69.7%	69.2%	69.4%	–	ACE2007
TimexTag	92.9%	81.3%	86.7%	–	91%	88.7%	89.9%	–	TERN2004
TEA	–	–	–	–	–	–	–	76.3%	Email Type
Ref. TDChoosing	–	–	–	–	89.6%	88.7%	89.2%	–	NewsWire
Bootstrapping	71.0%	52.8%	60.6%	–	–	–	–	–	ACE2005
KUL	85.0%	84.0%	84.5%	–	–	–	–	55.0%	TempEval-2
HeidelTime	90.0%	82.0%	86.0%	–	–	–	–	85.0%	TempEval-2

All the considered systems where evaluated on English news corpora, except the TEA system that was evaluated using a corpus of emails, and the Reference Time Dynamic Choosing system that used a Chinese news corpus. The TempEx system was evaluated comparing the system results with 221 news articles containing 728 timexes, Chronos with around 50000 words of

English news, TERSEO with 1821 broadcast news and newswire documents, DANTE was evaluated with a dataset containing approximately 70,000 words from news like documents, TimexTag with 511 newswire and broadcast news containing 5326 timexes, TEA with approximately 15000 emails, the Reference Time Dynamic-Choosing with 3148 Chinese news articles, the Bootstrapping system with 204.000 newswire documents and finally the KUL and HeidelTime systems was evaluated according to the Tempeval-2 evaluation data.

Looking at the table, the first important conclusion is the fact that all the systems, at least the ones that provide evaluation for the recognition task, have worst performance in the normalization task. This comes with no surprise, as seen from the previous sections, this fact is mainly due to the inherent difficulty to normalize temporal values.

The TEA system cannot be directly compared to the other systems because the domain, in which this system acts, i.e., emails, is very different from the other systems, which rely mainly on news text corpus.

Also, if we consider the method each system used, i.e., hand-crafted rules or machine learning, we see that the TempEx, Chronos, TERSEO and DANTE that are hand-crafted based systems, had, surprisingly, worst performance in the normalization task, although, this is mainly due to the fact that TimexTag recognizes less timexes than, for instance, Chronos.

Among the machine learning systems, we can see that several features used to train the models are more effective than other, for instance, POS tags, lexicons, and word patterns, are very effective. Others like suffixes and prefixes of the words, are less effective, depending on the systems.

Finally, from the systems described, we cannot infer what type of system, i.e. rule-based or machine learning, is better for resolving temporal expressions. The different type of datasets used by each system, the way the evaluation is done, are important factors to reach this conclusion.

Chapter 4

Using Machine Learning for Temporal Reference Resolution

Temporal reference resolution refers to identify and disambiguate the occurrences of references to temporal periods, given over textual documents.

In our work, we follow the TIMEX2 standard for defining what a temporal expression is and how one should be disambiguated. However, we focus mostly on the VAL attribute, ignoring the remaining TIMEX2 attribute information.

The proposed method for resolving temporal references in text is an instance of stacked learning, a machine learning paradigm that suggests constructing a combined model from several simpler learners (Wolpert, 1992). The basic concept behind stacking is to train two or more learners sequentially, with each successive learner incorporating the results of the previous one(s) in some fashion.

Figure 4.14 provides an illustration for the general stacked learning procedure. A first level learner corresponds to a Conditional Random Fields (CRF) tagger for identifying temporal references in text, based on the implementation available on the LingPipe package. This tagger annotates tokens as either being part of a reference to a specific point in time, a reference to a duration, a reference to a recurring period, a reference to a vague period in the past, present or future, a temporal reference of some other miscellaneous type, or some other class not corresponding to a temporal reference. Afterwards, a second learner takes as input the temporal references identified in the first level, and uses a ranking model to order and choose between a set of possible disambiguations obtained through a generative module. This module takes as input the reference recognized in the text and uses a small set of rules to generate a set of possible

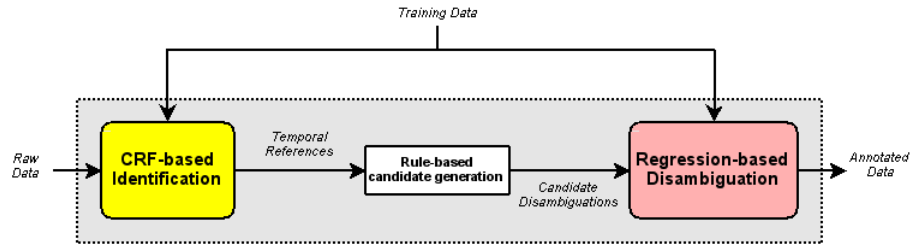

Figure 4.14: Temporal reference resolution with stacked learning.

disambiguation candidates. Each of these disambiguation candidates contains the values for the VAL attribute of the TIMEX2 annotation. The ranking module is trained to choose, from the set of candidates, the one whose temporal period denoted by the VAL attribute has the largest overlap with the true period of time corresponding to the temporal expression. The rest of this chapter details the learning approaches used in both levels.

4.1 Temporal Reference Identification

The first level learner of the proposed method is responsible for delimiting the occurrences of temporal references in the text. This task is addressed by translating what is essentially a chunking problem (i.e., a task of discovering the chunks of text that correspond to temporal references) into a tagging problem (i.e., a classification task of assigning tags to each individual text token, according to their belonging to a temporal reference or not), using the B-I-O encoding of chunkings as taggings. In our case, we addressed the chunking problem through the formulation of Conditional Random Fields, and we separately considered seven different types of temporal reference chunks:

- **Recurrences :** These chunks correspond to temporal expressions having TIMEX2 annotations where the SET attribute has a value of TRUE.
- **Durations :** These chunks correspond to expressions having TIMEX2 annotations where the SET attribute is empty and where the VAL attribute begins with either P or PT.
- **Points :** These chunks can assume one of two different possibilities, namely (i) a time-of-day without an associated date expression (i.e., the VAL attribute begins with T[0-9]), or (ii) date expressions of any granularity, from millennium down to hundredths of a second. The placeholder character, ”X”, can be used when parts of the value are unknown. In both cases, the TIMEX2 SET attribute is empty.

- **Ambiguous past references :** These chunks correspond to expressions having TIMEX2 annotations where the SET attribute is empty and where the VAL attribute assumes the vague token *PAST REF*. They do not require any further disambiguation.
- **Ambiguous present references :** These chunks correspond to expressions having TIMEX2 annotations where the SET attribute is empty and where the VAL attribute assumes the vague token *PRESENT REF*. They do not require any further disambiguation.
- **Ambiguous future references :** These chunks correspond to expressions having TIMEX2 annotations where the SET attribute is empty and where the VAL attribute assumes the vague token *FUTURE REF*. They do not require any further disambiguation.
- **Miscellaneous :** TIMEX2 annotations where the VAL attribute is empty, not requiring the disambiguation step.

The tagging problem involves separate B and I tags for each of the the seven different types of chunks. The classification model tags words given in a sequence according to the above tags, from which the system then generates the final results for the temporal reference recognition.

The identification process starts with a tokenization scheme that deterministically breaks an input text into a sequence of tokens. Tagging these tokens is nonetheless a non-trivial classification problem, since if there are K different tags, then a sequence of length N has up to K^N possible sequences of tags (although some may be eliminated with basis on structural grounds). A discriminative model known as Conditional Random Fields offers an efficient and principled approach for addressing this sequence classification problem (Lafferty *et al.*, 2001).

In this study, we used LingPipe's implementation of first-order chain conditional random fields (CRFs), which are essentially undirected probabilistic graphical models (i.e., Markov networks) in which vertexes represent random variables and each edges represent a dependency between two variables, that are discriminatively-trained to maximize the conditional probability of a set of hidden tags $y =< y_1, \ldots, y_C >$ given a set of input tokens $x =< x_1, \ldots, x_C >$. This conditional distribution has the following form:

$$p_\Lambda(y|x) = \frac{1}{\sum_y \prod_{c=1}^{C} \phi_c(y_c, x_c; \Lambda)} \prod_{c=1}^{C} \phi_c(y_{c-1}, y_c, x_c, c; \Lambda) \tag{4.1}$$

In the equation 4.1, ϕ are potential functions parametrized by Λ. Assuming ϕ_c factorizes a log-linear combination of arbitrary features computed over the subsequence c, then $\phi_c(y_{c-1}, y_c, x_c, c; \Lambda)$ $= exp(\sum_k \lambda_k f_k(y_{c-1}, y_c, x_c, c))$ where f is a set of arbitrary feature functions over the input, each of which having an associate model parameter λ_k. The feature functions can informally

be thought of as measurements on the input sequence that partially determine the likelihood of each possible value for y_c. The parameter k represents the number of considered features and parameters $\Lambda = \{\lambda_k\}$ are a set of real-valued weights, typically estimated from labeled training data by maximizing the data likelihood function through stochastic gradient descent. Given a CRF model, finding the most probable sequence of hidden tags given some observations can be made through the Viterbi algorithm, a form of dynamic programming applicable to sequence tagging. LingPipe's implementation makes the first-order Markov assumption on the dependencies among y (i.e., the transitions between tags y depend only on the origin and destination), and thus, at sequence position c, feature functions only see y_c and y_c corresponding to first-order chain CRFs.

The modeling flexibility of CRFs permits the feature functions to be complex, overlapping features of the input, without requiring additional assumptions on their inter-dependencies. The list of features considered in our experiments includes:

- Context words: The token identity within a 1-token window of the target token.
- Lexicons: Whether the token appears in a list of bare weekday names, month names, or specific words associated with temporal expressions.
- Regular expressions: Whether the token is capitalized, contains punctuation of specific digit patterns.
- Part-Of-Speech tags: Predicted by an HMM that was trained separately for English part-of-speech tagging using the Brown corpus.
- Prefix-suffix: The four letter prefix and suffix of the word.

4.2 Temporal Reference Disambiguation

Our approach for temporal reference disambiguation involves (i) generating disambiguation candidates by using a generative approach based on rules, (ii) scoring possible candidates using an SVM regression model, and (iii) selecting the highest scoring candidate. The regression objective used in the second step is based on the overlap between the true temporal period associated with the expression and the temporal periods of the candidates. Figure 4.15 provides an illustration for the general disambiguation procedure.

The proposed disambiguation module uses class-specific rules to compute, for each temporal reference, a set of possible candidate disambiguations. Each rule consists of a regular expression pattern, which may refer to a small lexicon of names, units, and numeric words, and a

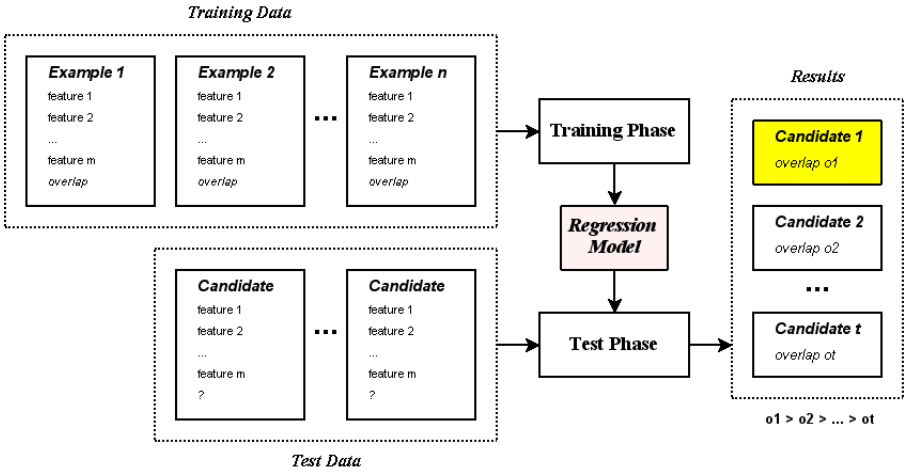

Figure 4.15: Temporal reference disambiguation through regression.

generator that produces values for the TIMEX2 VAL attribute. Rules are applied by evaluating the rule's pattern against the text of the temporal expression. In case of a successful match, a candidate disambiguation is produced. In total, we consider a set of 21 rules. Table 4.6 gives the distribution of rules for the classes of temporal expressions and presents some example rules. In the table, tokens in `ALLCAPS` indicate lexical classes, tokens in `MixedCase` indicate other rules, and tokens in `lowercase` indicate lexical items.

It should be noticed that while some temporal expressions of the class point are fully qualified, others require a reference time (i.e., a temporal anchor) to be fully disambiguated. Some temporal expressions, such as *yesterday*, *two years ago*, or *next month*, are deictic and should be anchored to the time of speech (e.g., a document timestamp). Others, such as *three days earlier* or *the next year*, are anaphoric and should be anchored to a salient temporal expression in the same discourse unit. A temporal expression may also contain its own anchor, such as in the case of *three days after September 11*, whose anchor is the embedded anaphoric expression *September 11*. The code associated with the rules takes as input, besides the text of the reference, a document timestamp and the VAL attributes of all candidate temporal expressions computed earlier in that document, producing a set of candidate disambiguations for the temporal expression being computed. The VAL attributes of the candidates are always represented as intervals, and, by doing so, we normalize all the temporal references. For this purpose we use the JodaTime Java API[1], which allows us to easily manipulate time expressions. Time points expressions (i.e., temporal instances) can be easily represented by intervals, if we consider the granularity of that time expression. If the granularity is, for instance, month, we represent that by an interval that

[1] http://joda-time.sourceforge.net/

Class	Rules	Example
Duration	5	(around|about|roughly|nearly|over) (TEXTUAL_NUMBER|NUMERIC_DAYS) CALENDAR_GRANULARITY
		the (first|last) TEXTUAL_NUMBER CALENDAR_GRANULARITY of (YEAR|MONTHSPEC)
		(next|previous|last|following|a) (few|many|TEXTUAL_NUMBER|NUMERIC_DAYS) CALENDAR_GRANULARITY
		(the)?((latest)|(per)|(past)|(under)|(over)| (near)|(next)|(previous)|(last)|(following)|(a))? ((few)|(many)|(TEXTUAL_NUMBER)|(NUMERIC_DAYS))? (of that)? (CALENDAR_GRANULARITY) (long|old)?
Point	16	(NUMERIC_DAYS | MONTHSPEC).? (of)? (YEAR)?
		DAYSPEC? ?Numeric31 (Numeric31|MONTHSPEC) YearFourDigitOrYearTwoDigits
		MONTHSPEC.? Numeric31(st|nd|rd|th)?
		(DAYSPEC)?Numeric31 (Numeric31|MONTHSPEC) YEAR

Table 4.6: Rules for generating candidate disambiguations.

goes from the beginning of the month to the ending of that month. For the case of TIMEX2 durations, we represent them by temporal intervals starting at midnight UTC of January the 1st of 1970, and spanning the duration associated to the temporal expression.

After generating disambiguation candidates, and for each pair $<$*temporal-reference*, *candidate-disambiguation*$>$, the following set of features is computed:

- The number of milliseconds between the center point of the candidate temporal disambiguation and the document timestamp.
- The number of milliseconds between the center point of the candidate disambiguation and the closest (in terms of number of elapsed milliseconds towards the center point) candidate disambiguation of other temporal expression recognized in the same document.
- The number of milliseconds between the center point of the candidate disambiguation and the closest (in terms of number of elapsed milliseconds towards the center point) candidate disambiguation of other temporal expression recognized in the same sentence.
- The maximum duration (in terms of milliseconds) of the period of temporal overlap between the candidate temporal disambiguation and some other candidate temporal disambiguation corresponding to a temporal reference recognized in the same document.
- The maximum duration (in terms of milliseconds) of the period of temporal overlap between the candidate temporal disambiguation and some other candidate temporal disambiguation corresponding to a temporal reference recognized in the same sentence.

- A numeric value indicating if the temporal period denoted by the VAL attribute corresponds to a duration of (1) less then a minute, (2) hour, (3) day, (4) week, (5) month, (6) year, (7) decade, (8) century, or (9) millennium.
- The number of milliseconds from midnight UTC of January 1, 1970, to the beginning of the temporal interval corresponding to the candidate.
- The number of milliseconds from midnight UTC of January 1, 1970, to the end of the temporal interval corresponding to the candidate.
- The duration, in milliseconds, of the candidate's interval.
- A set of features encoding the occurrence of particular textual terms in the text of the temporal expression, through a binary value. The idea behind these features is to have, in the model, a lexic of the terms used to represent different temporal expressions.

When computing the above features, the temporal axis is assumed to start at the midnight Coordinated Universal Time (UTC) of January 1, 1970, and end at 03:14:07 UTC of Tuesday, the 19th of January in 2038, which is the default temporal axis in the JodaTime API that we use to compute distances and overlaps for the different types of temporal expressions. We are assuming that the temporal references that we are dealing with always correspond to periods after 1970, although this could easily be changed to some other date in the past.

For training and evaluating the SVM regression model, we also compute the temporal overlap between each candidate and the true temporal period associated with the reference, as described in the golden collection. This overlap is computed by dividing the number of milliseconds for the period of overlap between the two intervals, by the number of milliseconds between the beginning of the first interval and the ending of the second interval, so that we can have a normalized value. The regression model attempts to estimate this overlap, with basis on the considered features. More formally, let X be the training set consisting of m instance pairs $x_i = \{f_{i,1}, \ldots, f_{i,n}, d_i\}$ where $f_{i,j}$ is the j-th input feature of a given example i and d_i is the corresponding temporal overlap. The regression goal is to estimate a function $reg(x)$ with basis on the training set X that takes as input a set of instance pairs $x'_i = \{f_{i,1}, \ldots, f_{i,n}\}$, is as close as possible to the target values d_i for every x'_i, and at the same time is as flat as possible for good generalization.

Support Vector Machines (SVMs) are a set of machine learning approaches used for classification and regression, developed in the mid 90's by Vapnik and his co-workers at AT&T Bell Labs Kolomiyets & Moens (2009). In the case of SVM regression, the basic idea is to map the features into a high-dimensional feature space via a kernel function (which may be linear or not),

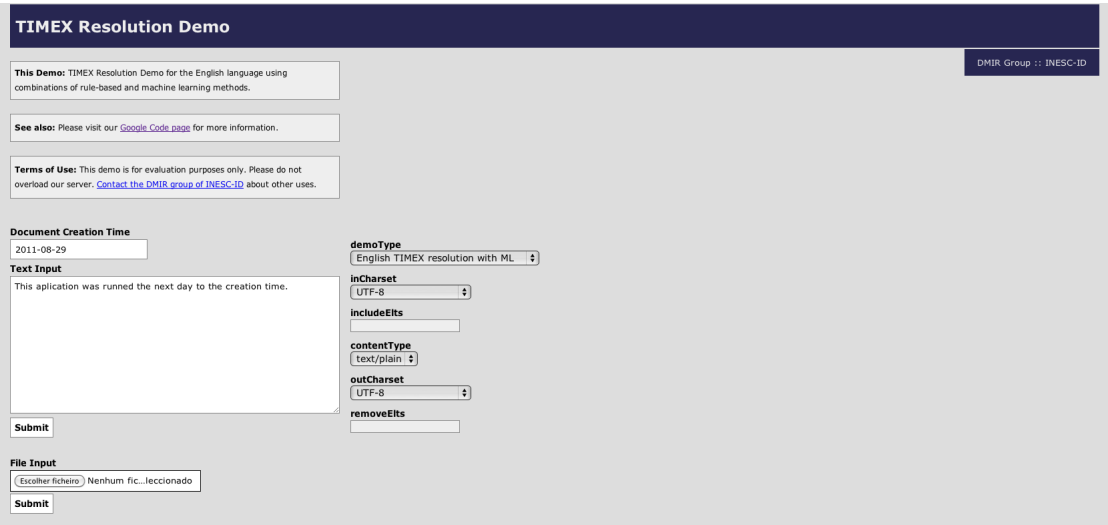

Figure 4.16: Image of the Web Application

and then do a linear regression in this new space. Many different algorithms have been proposed for training SVM regression models. The Weka machine learning toolkit, which we used in this work, implements the approach proposed by Shevade et al., which in turn is an improvement over Smola and Scholkopf's sequential minimal optimization (SMO) algorithm for training a support vector regression (Mani & Wilson, 2000). In our experiments, we used Weka's implementation with a Radial Basis Function (RBF) as kernel. The parameters C and γ were set to the default values of 250007 and 0.01 respectively.

The final step of the proposed approach involves selecting the candidate with the largest estimated overlap as the result for the disambiguation.

4.3 Prototype System Implementing the Proposed Method

To demonstrate the system, a Google App Engine application was made available under the name stemptag[1].

As we can see from Figure 4.16 the user is invited to test the application in two ways. The first way is to enter the text, to which the temporal expressions are to be resolved, in a text box, as shown in the figure. The second way is to submit a file to the application by using the *File Input* option, and uploading a file. Then in the first drop box on the right, the user has several options. The first option is to detect temporal expressions using rules, the second option is to resolve

[1] `http://stemptag.appspot.com/`

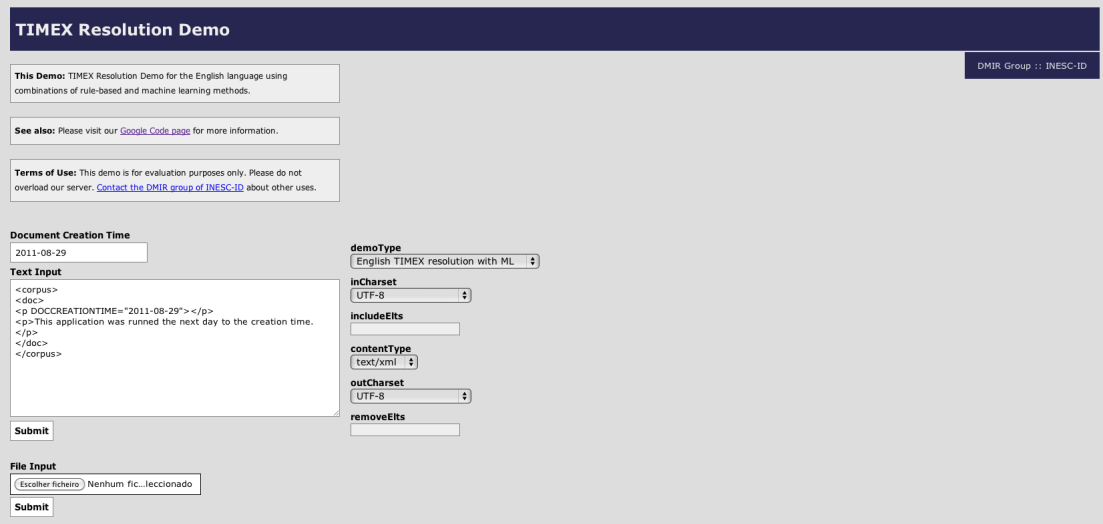

Figure 4.17: Image of the Web Application using XML

the temporal expressions using machine learning, this option uses the method described in this dissertation, the third option shows the POS tags of the text, this POS tags are used in the CRF model described earlier, and the fourth option splits the text into sentences.

Figure 4.16 shows an example of usage, were the user enters a text in the text box, selects the second option on the right, which means resolve the temporal expressions using machine learning, and then presses the *Submit* option, which outputs the result in a XML form shown in Figure 4.1.

```
<output>
    <s i="0">
        This aplication was runned <TIMEX2 val="2011-08-30">the next day</TIMEX2> to the creation
            time.
    </s>
</output>
```

Listing 4.1: Output of the Web Application

Other options are presented in the application as well, the second dropbox on the right is used to choose the charset to be used in the input of the application, the third dropbox is used to choose the content type of the text submitted to the application, in th example the format is plain text, but the application also admits text in XML with a specific format, as shown in Figure 4.17. The last dropbox is used to choose the charset of the output of the application. This instructions are also present in the application page.

The source code of the application was also made available in a Google Code page under the

name stemptag[1].

4.4 Summary

In this section a novel method using machine learning was described.

This method uses both machine learning in the recognition and in the disambiguation task of the temporal expression resolution problem. The recognition module uses a CRF model to recognize the extent of the temporal expressions present in the text, and the normalization module uses a SVM regression module to fill the *VAL* attribute. Between these two modules, another, based only on 21 rules, is used to produce possible disambiguation candidates to the SVM regression module. The architecture of the system and the features used in the machine learning modules are also described.

Finally, a web demonstration of the system is also described, as well as the Google Code page were the source code was made available, showing screenshots of the application and how it works.

[1] http://code.google.com/p/stemptag/

Chapter 5

Experimental Validation

In this section, we describe the details of our empirical evaluation. This includes the details of our experimental design (i.e., the datasets and the evaluation metrics that were used to measure the quality of the results), as well as results for the experiments that evaluated the effectiveness of the methods under study.

5.1 Evaluation Methodology

Our validation methodology is based on the standard NER evaluation setup, which involves comparing the annotations produced by the automated system against gold-standard annotations provided by human experts. As gold-standard annotated data, we used one English datasets with TIMEX2 annotations, namely WikiWars, and two English dataset with TimeML annotations, namely TimeBank and the AQUAINT TimeML corpus. In the two last dataset, the TIMEX3 annotations from the TimeML standard were converted to TIMEX2 annotations using a set of rules inspired in the work of Saquete (2010). Section 2.2 briefly described the TIMEX2 annotation standard and Table 5.7 presents a statistical characterization for these collections, also indicating the state-of-the-art results that have been reported for each. Each document in these collections represents an article formatted in XML, in which TIMEX2 tags denote temporal expressions. For the purpose of our evaluation we trained our models using 80% of each dataset, and tested the system with the remaining 20%.

Given our two-stage approach, we took separate measurements for the identification and disambiguation sub-tasks. In both these cases, we measured results with the commonly used F_1 rate, which is equal to the harmonic mean between precision and recall. Precision is the percentage of

		TimeBank	AQUAINT	WikiWars
	Num. documents	186	73	22
	Num. words	68.500	34.154	120.000
	Discourse domain	news	news	news
	Number of temporal expressions	1423	605	2671
	Expressions of type recurrence	20	14	55
	Expressions of type point	975	482	2112
	Expressions of type duration	238	86	247
	Expressions of type ambiguous present	73	15	100
	Expressions of type ambiguous past	68	0	53
	Expressions of type ambiguous future	40	8	44
	Expressions of type miscellaneous	10	0	60
	F_1 recognition	0.86	0.93	0.95
Best Results	F_1 disambiguation	NA	0.84	0.58
	Accuracy disambiguation	0.85	NA	NA

Table 5.7: The datasets used in our validation experiments.

correct references identified and/or disambiguated by the system. Recall is the percentage of references present in the test collection that are identified/disambiguated by the system. A temporal reference is correctly identified only of it is an exact match with the corresponding temporal reference given in the test collection, and correctly disambiguated only if the assigned disambiguation is an exact match with the one that is given in the TIMEX2 annotation (i.e., the VAL attribute has exactly the same value). Thus, correct disambiguation can only occur when a correct recognition has first taken place, since it does not make sense to disambiguate textual expressions that are not recognized as temporal references. However, previous results were not always computed this way, and this can be a source of problems when comparing results. Systems that do not use this way to produce results, can achieve a better result, although being a worst system.

5.2 The Obtained Results

We compared the proposed approach using different features for the CRF temporal expression identification model. Section 4.1 described the complete set of features that we considered and Table 5.8 presents the obtained results across the three different document collections. In Table 5.9 we can see the results obtained by the CRF model that considered the full set of features, in all temporal reference chunk types. Results are shown in terms of precision (P), recall (R) and F_1.

The obtained results vary between 0.64 - 0.93 in F_1 in the recognition sub-task and 0.31 - 0.67 in F_1 in the disambiguation sub-task. This results are comparable, and in many cases favorably, against those that have been previously reported over the same datasets, see Table 5.7.

In terms of disambiguation accuracy, we separately measured the quality of the produced disambiguations for two different classes of ambiguous temporal references, namely durations and

	WikiWars			AQUAINT			TimeBank		
	P	R	F_1	P	R	F_1	P	R	F_1
Context words	**0.93**	0.86	0.90	0.86	0.64	0.73	0.79	0.58	0.67
Context words and rules	**0.93**	0.87	0.90	**0.87**	0.69	0.77	**0.80**	0.59	0.68
Context words and lexicons	**0.93**	0.87	0.90	0.84	0.80	0.82	0.77	0.60	0.67
Context words and POS	0.92	0.86	0.89	0.84	0.65	0.73	0.77	0.56	0.64
Context words and Pref_Suf	0.89	0.86	0.87	0.78	0.65	0.71	0.75	**0.63**	**0.69**
Context words, rules and lexic.	0.91	0.87	0.89	0.86	0.83	**0.84**	0.79	0.59	0.68
Context words, rules and POS	**0.93**	0.87	0.90	0.86	0.69	0.76	**0.80**	0.55	0.65
Context words, rules and Pref_Suf	0.89	0.86	0.87	0.78	0.66	0.72	0.76	0.61	0.68
Context words, lexicons and POS	**0.93**	**0.92**	**0.93**	0.83	**0.84**	0.83	0.78	0.60	0.68
All features	0.90	0.88	0.89	0.81	0.81	0.81	0.76	**0.63**	0.68

Table 5.8: The obtained results when comparing different recognition models.

	WikiWars			AQUAINT			TimeBank		
	P	R	F_1	P	R	F_1	P	R	F_1
Recurrences	1.00	0.71	0.83	0.0	0.0	0.0	0.0	0.0	0.0
Points	0.78	0.92	0.84	0.82	0.88	0.85	0.44	0.75	0.55
Durations	0.74	0.50	0.60	0.50	0.30	0.38	0.79	0.25	0.38
Ambiguous past refs.	1.00	0.46	0.63	0.0	0.0	0.0	0.75	0.38	0.55
Ambiguous present refs.	0.72	0.77	0.74	0.0	0.0	0.0	0.0	0.0	0.0
Ambiguous future refs.	1.00	0.25	0.40	0.0	0.0	0.0	0.0	0.0	0.0
Miscellaneous	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Table 5.9: The obtained recognition results for the different classes that were considered.

points. In these experiments, we performed the recognition through the CRF model considering the entire set of features.

5.3 Discussion

The results show that, as expected, the CRF tagger using a richer feature set achieves the best results. Prefix and Suffix features seem to be the least informative. This can be due to the many different prefixes and suffixes that temporal expressions can have, not corresponding to highly discriminative specific patterns.

Table 5.10 presents the obtained results for the disambiguation sub-task, showing that, overall, the proposed approach is comparable or even outperforms other state-of-the-art systems. Although the values shown in Table 5.7 for the state-of-the-art disambiguation performance appear

	WikiWars			AQUAINT			TimeBank		
	P	R	F_1	P	R	F_1	P	R	F_1
Durations	0.72	0.23	0.35	0.60	0.27	0.38	0.78	0.16	0.27
Points	0.57	0.54	0.56	0.64	0.53	0.58	0.38	0.26	0.31
Overall	0.69	0.65	0.67	0.71	0.57	0.63	0.38	0.27	0.31

Table 5.10: The obtained results for the disambiguation sub-task.

to be better, we have that these previous works reported results for temporal expression disambiguation separately from the recognition, whereas we considered failures in the recognition as failures also in terms of disambiguation. Moreover, we were more restricted in the types of temporal expressions considered for disambiguation. Thus, results are not directly comparable.

The values of zero shown in Table 5.9, associated with some particular types of temporal expressions, were obtained due to an insufficient amount of training data corresponding to those kinds of expressions, see Table 5.7. In the TempEval-2 challenge that used the TimeBank dataset, the average F1 score for temporal expression recognition was of 0.61 and, if we remove the two best systems, the average is only 0.56, which is clearly bellow our system performance. It is also worth noticing that the best results reported for the datasets considered here were achieved with ruled-based approaches in the normalization subtask, which are harder to develop and maintain.

A manual inspection on the produced annotations revealed that some of the frequent errors in the produced annotations correspond to problems such as those described bellow:

- There were several legitimate temporal expressions that were recognized by our method and missing from the gold-standard annotations. For example our method recognized, in a sentence, the time expression *"early March"*, while in the gold-standard annotation only *"March"* was tagged in the same sentence, leading to a decrease in the results. Other authors have also reported similar problems (Mazur & Dale, 2007).
- A significant percentage of the recognition errors were related to expressions which contained numbers wrongly recognized as years, dates or hours.
- Several gold-standard annotations were wrong for both point and duration time expressions. For example, expressions like *"now"*, were sometimes annotated with the val attribute PRESENT_REF and other times with a normalized time expression like *"2011-10-10"*.

Despite the above problems, it is our belief that the achieved results are of an acceptable quality to be used in the subsequent processing stages of many different types of Temporal Information Retrieval applications. Our currently ongoing work is pursuing this path, particularly focusing on text analytics and information retrieval methods that combine temporal with geospatial text annotations (**?**).

Chapter 6

Conclusions

This work proposed a supervised machine learning method for resolving temporal references in text. The proposed method is an instance of stacked learning (Wolpert, 1992), in which a first learner based on Conditional Random Fields (CRFs) is used to tag temporal references, and then a second learner based on SVM regression is used to rank the possible candidate disambiguations for the temporal references that were initially tagged. Experiments with English datasets containing TIMEX2 and TimeML annotations attested for the adequacy of the proposed approach, showing that it compares well against previous methods.

In short, the system begins by receiving a document that is first passed through a CRF model, which allows the system to recognize the extent of the temporal expressions present in the text. The system then passes each of the recognized temporal expressions to a rule-based module, with only 21 rules, producing possible candidate disambiguations, which are in turn passed to the SVM regression module that ranks the candidates and chooses the the best candidate to disambiguate the temporal expression. Finally, the system outputs the results in a XML format.

The obtained results vary between 0.64 - 0.93 F_1 in the recognition sub-task and 0.31 - 0.67 F_1 in the disambiguation sub-task, showing that, overall, the system is well suited for the temporal expressions resolution problem.

6.1 Future Work

Despite the interesting results, there are also many challenges for future work. While the ACE TERN initiative along with the TIMEX2 annotation standard addressed the recognition and normalization problems, more advanced temporal processing was not covered. The most recent

annotation language for temporal expressions, TimeML, opens up new horizons for automated temporal information extraction and reasoning, by considering things like the time stamping and ordering of events. For future work, it would be interesting to extend the methodology proposed in this paper in order to address the annotation of documents according to the complete TimeML standard, since for now we only transformed the TimeML annotations into the format of TIMEX2.

Extend the system to recognize several languages, namely Portuguese, since for now we only recognize English language.

Another point to the future is to resolve the complete set of attributes present in the TIMEX2 format, as well as recognize the relations between each temporal expression. Also, using more advanced machine learning models, namely second-order CRFs to better recognize temporal expressions, as well as trying unsupervised and semi-supervised techniques.

Resolving geospatial references over text also has important applications and equally challenging problems (Martins *et al.*, 2010). Some work concerning this problem was already done (Loureiro *et al.*, 2011a), the system basically uses the same method described in this dissertation to resolve both temporal and geospatial expressions.

Bibliography

ABNEY, S. (1996). *Part-of-Speech Tagging and Partial Parsing*. Kluwer Academic.

AHN, D., VAN RANTWIJK, J. & DE RIJKE, M. (2007). A cascaded machine learning approach to interpreting temporal expressions. In *Proceedings of Annual Conference of the North American Chapter of the Association for Computational Linguistics*.

AHO, A.V. & CORASICK, M.J. (1975). Efficient string matching: An aid to bibliographic search. *Communications of the ACM*, **18(6)**.

BAUM, L.E., PETRIE, T., SOULES, G. & WEISS, N. (1970). A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains. *The Annals of Mathematical Statistics*, **41(1)**.

BENDER, O., OCH, F. & NEY, H. (2003). Maximum entropy models for named entity recognition. In *Proceedings of the 7th Conference on Natural Language Learning*.

CARPENTER, B. & BALDWIN, B. (2011). *Natural Language Processing with LingPipe 4*. LingPipe Publishing.

C.FELLBAUM (1998). *WordNet, An Electronic Lexical Database*. The MIT Press.

CHANG, C. & LIN, C. (2001). *LIBSVM: a library for support vector machines*.

CHARNIAK, E. (2000). A maximum-entropy-inspired parser. *Proceedings of the 1st Conference of the North American chapter of the Association for Computational Linguistics*.

CHIEU, H. & NG, H. (2003). Named entity recognition with a maximum entropy approach. In *Proceedings of the 7th Conference on Natural Language Learning*.

CUNNINGHAM, H., MAYNARD, D., BONTCHEVA, K. & TABLAN, V. (2002). GATE: an architecture for development of robust hlt applications. *Proceedings of the 40th Anniversary Meeting of the ACL*.

DESCHACHT, K. & MOENS, M.F. (2009). Semi-supervised semantic role labeling using the latent words language model. *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*.

DUGAD, R. & DESAI, U.B. (1996). A tutorial on Hidden Markov Models. Tech. Rep. SPANN-96.1, Indian Institute of technology.

FLORIAN, R., ITTYCHERIAH, A., JING, H. & ZHANG, T. (2003). Named entity recognition through classifier combination. In *Proceedings of the 7th Conference on Natural Language Learning*.

HAN, B. (2003). Time calculus for natural language: Tagging guidelines, unpublished draft.

HAN, B. & LEVIN, L. (2006). Understanding temporal expressions in emails. *Proceedings of the Human Language Technology Conference*.

HAN, B., GATES, D. & LEVIN, L. (2006). From language to time: A temporal expression anchorer. *In Proceedings of the 13th International Symposium on Temporal Representation and Reasoning*.

JURAFSKY, D. & MARTIN, J.H. (2000). *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice-Hall.

KOLOMIYETS, O. & MOENS, M.F. (2009). Comparing two approaches for the recognition of temporal expressions. In *Proceedings of the 32nd Annual German Conference on Artificial Intelligence*.

KOLOMIYETS, O. & MOENS, M.F. (2010). Kul: recognition and normalization of temporal expressions. *Proceedings of SemEval-2 5th workshop on semantic evaluation*.

KRAUT, R.E., FUSSELL, S.R., LERCH, F.J. & ESPINOSA, A. (2005). Coordination in teams: Evidence from a simulated management game. *Journal of Organizational Behavior*.

LAFFERTY, J., MCCALLUM, A. & PEREIRA, F. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning*.

LOUREIRO, V., ANASTÁCIO, I. & MARTINS, B. (2011a). Learning to resolve geographical and temporal references in text. *Proceeding of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*.

LOUREIRO, V., MARTINS, B. & CALADO, P. (2011b). A machine learning method for resolving temporal references in text. *Proceedings of the 15th Portuguese Conference on Artificial Intelligence*.

MANI, I. & WILSON, G. (2000). Robust temporal processing of news. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*.

MARTINS, B., ANASTÁCIO, I. & CALADO, P. (2010). A machine learning approach for resolving place references in text. *Proceedings of the 13th AGILE International Conference on Geographic Information Science*.

MAYFIELD, J., MCNAMEE, P. & PIATKO, C. (2003). Named entity recognition using hundreds of thousands of features. In *Proceedings of the 7th Conference on Natural Language Learning*.

MAYNARD, D., TABLAN, V., URSU, C., CUNNINGHAM, H. & WILKS, Y. (2001). Named entity recognition from diverse text types. *Proceedings of the Conference on Recent Advances in Natural Language Processing*.

MAZUR, P. & DALE, R. (2007). A rule based approach to temporal expression tagging. In *Proceedings of the International Multiconference on Computer Science and Information Technology*.

MCCALLUM, A. & LI, W. (2003). Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In *Proceedings of the 7th Conference on Natural Language Learning*.

N., D. & SEKINE, S. (2007). A survey of named entity recognition and classification. *Linguisticae Investigationes*, **1(30)**.

NEGRI, M. & MARSEGLIA, L. (2004). Recognition and normalization of time expressions: ITC-irst at TERN 2004. Tech. Rep. WP3.7 – MEANING Project Deliverable, ITC-irst, Trento.

POVEDA, J., SURDEANU, M. & TURMO, J. (2009). An analysis of bootstrapping for the recognition of temporal expressions. *Proceedings of the NAACL HLT 2009 Workshop on Semi-Supervised Learning for Natural Language Processing*.

PUSTEJOVSKY, J., CASTAÑO, J., INGRIA, R., SAURÍ, R., GAIZAUSKAS, R., SETZER, A. & KATZ, G. (2003). TimeML: Robust specification of event and temporal expressions in text. In *Proceedings of the 5th International Workshop on Computational Semantics*.

SANG, E. & MEULDER, F.D. (2003). Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the 7th Conference on Natural Language Learning*.

SAQUETE, E. (2010). Terseo + t2t3 transducer: a systems for recognizing and normalizing timex3. *Proceedings of the 5th International Workshop on Semantic Evaluation*.

SAQUETE, E., MUÑOZ, R. & MARTÍNEZ-BARCO, P. (2003). Terseo: Temporal expression resolution system applied to event ordering. *Proceedings of the 6th International Conference on Text, Speech and Dialogue*.

SAQUETE, E., MUÑOZ, R. & MARTÍNEZ-BARCO, P. (2005). Event ordering using the TERSEO system. *Data and Knowledge Engineering*, **58(1)**.

STRÖTGEN, J. & GERTZ, M. (2010). Heideltime: High quality rule-based extraction and normalization of temporal expressions. *Proceedings of the 5th International Workshop on Semantic EvaluationProceedings of the 5th International Workshop on Semantic Evaluation*.

VERHAGEN, M. & MOSZKOWICZ, J.L. (2009). Temporal annotation and representation. *Language and Linguistics Compass*, **3(2)**.

VITERBI, A. (1967). Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, **13(2)**.

WILSON, G., MANI, I., SUNDHEIM, B. & FERRO, L. (2001). A multilingual approach to annotating and extracting temporal information. *Proceedings of the Workshop on Temporal and Spacial Information Processing*, a Multilingual Approach to Annotating and Extracting Temporal Information.

WITTEN, I. & FRANK, E. (2000). *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann.

WOLPERT, D. (1992). Stacked generalization. *Neural Networks*, **5(2)**.

ZHAO, X., JIN, P. & YUE, L. (2010). Automatic temporal expression normalization with reference time dynamic-choosing. *Proceedings of the 23rd International Conference on Computational Linguistics*.

ZHOU, G. & SU, J. (2002). Named entity recognition using an HMM-based chunk tagger. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*.

Appendix A

Appendix A

```
<corpus>
<doc>
</doc>
<doc>
<p DOCCREATIONTIME="2009-12-19T17:00:00"></p>
<p>Course of the war</p>
<p></p>
<p>War breaks out in Europe</p>
<p></p>
<p>On <TIMEX2 val="1939-09-01" mod="" set="" anchor_val="" anchor_dir="">September 1,
    1939</TIMEX2>, Germany and Slovakia - a client state in <TIMEX2 val="1939" mod="" set=""
    anchor_val="" anchor_dir="">1939</TIMEX2> - attacked Poland and World War II broke out. France,
    Britain, and the countries of the Commonwealth declared war on Germany but provided little
    military support to Poland other than a small French attack into the Saarland. On <TIMEX2
    val="1939-09-17" mod="" set="" anchor_val="" anchor_dir="">September 17, 1939</TIMEX2>, after
    signing an armistice with Japan, the Soviets launched their own invasion of Poland. By <TIMEX2
    val="1939-10" mod="START">early October</TIMEX2>, Poland was divided among Germany, the Soviet
    Union, Lithuania (returned Vilnius capital province) and Slovakia, although Poland never
    officially surrendered and continued the fight outside its borders. At <TIMEX2 val="" mod=""
    set="" anchor_val="" anchor_dir="">the same time as the battle in Poland</TIMEX2>, Japan
    launched its first attack against Changsha, a strategically important Chinese city, but was
    repulsed by <TIMEX2 val="1938-09" mod="END">late September</TIMEX2>.</p>
<p></p>
<p>Following the invasion of Poland and a German-Soviet treaty governing Lithuania, the Soviet
    Union forced the Baltic countries to allow it to station Soviet troops in their countries under
    pacts of "mutual assistance." Finland rejected territorial demands and was invaded by the
    Soviet Union in <TIMEX2 val="1939-11" mod="" set="" anchor_val="" anchor_dir="">November
    1939</TIMEX2>. The resulting conflict ended in <TIMEX2 val="1940-03" mod="" set=""
    anchor_val="" anchor_dir="">March 1940</TIMEX2> with Finnish concessions. France and the United
    Kingdom, treating the Soviet attack on Finland as tantamount to entering the war on the side of
    the Germans, responded to the Soviet invasion by supporting its expulsion from the League of
    Nations. In <TIMEX2 val="1940-06" mod="" set="" anchor_val="" anchor_dir="">June 1940</TIMEX2>,
    the Soviet Armed Forces invaded and occupied the neutral Baltic States.</p>
<p></p>
<p>In Western Europe, British troops deployed to the Continent, but in a phase nicknamed the Phoney
    War by the British and "Sitzkrieg" by the Germans, neither side launched major operations
    against the other until <TIMEX2 val="1940-04" mod="" set="" anchor_val="" anchor_dir="">April
    1940</TIMEX2>. The Soviet Union and Germany entered a trade pact in <TIMEX2
    val="1940-02">February of 1940</TIMEX2>, pursuant to which the Soviets received German military
    and industrial equipment in exchange for supplying raw materials to Germany to help circumvent
    a British blockade. In <TIMEX2 val="1940-04">April</TIMEX2>, Germany invaded Denmark and Norway
    to secure shipments of iron ore from Sweden, which the allies would try to disrupt. Denmark
    immediately capitulated, and despite Allied support, Norway was conquered within <TIMEX2
    val="P2M" anchor_val="1940-04" anchor_dir="STARTING">two months</TIMEX2>. British discontent
    over the Norwegian campaign led to the replacement of Prime Minister Neville Chamberlain by
    Winston Churchill on <TIMEX2 val="1940-05-10" mod="" set="" anchor_val="" anchor_dir="">May 10,
    1940</TIMEX2>.</p>
```

Listing A.1: Extracted example from the WikiWars Train Corpus