A Machine Learning Method for Resolving Temporal References in Text

Vítor Loureiro

Instituto Superior Técnico, INESC-ID
Av. Professor Cavaco Silva, 2744-016 Porto Salvo, Portugal

Abstract. This paper 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 through English corpora containing TIMEX2 and TimeML annotations for the temporal references. The results show that the proposed method compares well against previously proposed approaches.

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 [7,2,15]. 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 [13,11]. 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

This work was partially supported by the Fundação para a Ciência e a Tecnologia (FCT), through project grant PTDC/EIA-EIA/109840/2009 (SInteliGIS)

used in the recognition subtask, very few studies have addressed their application to the disambiguation subtask.

This paper proposes a supervised machine learning method for resolving temporal references in text. The proposed method is an instance of stacked learning [17], 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. The second learner is perhaps the most innovative contribution of this paper.

In terms of implementation, the first learner is based on the CRF tagger from the LingPipe package [15]. The second learner is based on the SVM regression implementation available in the Weka machine learning toolkit [16], 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. The results show that machine learning methods are indeed effective and that they compare well against the state-of-the-art approaches.

The rest of this paper is organized as follows: Section 2 presents related works, covering both the areas of named entity recognition and temporal reference resolution. Section 3 presents the proposed machine learning method, detailing the individual learners that were used for temporal reference identification and for temporal reference resolution. Section 4 presents the experimental validation of the proposed method. Finally, Section 5 presents our conclusions and points directions for future work.

2 Concepts and Related Work

Temporal reference resolution can be divided into two separate sub-tasks, namely temporal reference identification and temporal reference disambiguation. Identification concerns with delimiting, in a document, the character strings that refer to different types of temporal expressions (e.g., recurring periods or specific time points and durations, that can be either exact or vague). This is a particular instance of the more general problem of Named Entity Recognition, which has been extensively studied in the Natural Language Processing (NLP) community [11]. Temporal reference disambiguation refers to associating the recognized references to the corresponding calendar instances or intervals. In this section, we survey relevant previous works for both the temporal reference identification and disambiguation sub-tasks.

2.1 Named Entity Recognition

The named entity recognition (NER) task, as proposed in the 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, 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%. Nadau and Sekine provide a recent survey in the area [11].

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 against a sequence of words in much the same way as a general regular expression matcher. 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 the 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 computed at either the document scale or 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. Several different classification methods have been successful applied on this task [13]. For instance Zhou and Su experimented with Hidden Markov Models (HMMs), also using a large variety of features [18]. Conditional Random Fields (CRFs), a generalization of ME and HMMs, were explored by McCallum and Li [10]. Florian et al. [4] combined Maximum Entropy and Hidden Markov Models under different conditions. Recent works have shown that CRFs have advantages over traditional HMMs in the face of many redundant features, although CRF models need fairly extensive feature engineering to outperform a good HMM baseline. The method reported in this paper is based on the CRF tagger implementation available from LingPipe, a suite of Java libraries for the analysis of human language. Section 3.1 details the CRF-based method used for temporal reference identification.

2.2 Temporal Reference Resolution

For most Temporal Information Retrieval applications, the handling of temporal references requires not only recognizing the mentions to time points or periods given over the text (i.e., delimiting their occurrences), but also disambiguating these occurrences into the corresponding intervals over a calendar. Having the temporal disambiguations is, for instance, crucial for supporting document analytics over the temporal dimension, or document visualizations with basis on timelines [3]. However, the annotation of temporal references is still a very difficult challenge for natural language processing, requiring additional steps besides the simple normalization of date expressions. Time consists of more than calendar dates and clock times, as it also includes points of finer and coarser granularity, durations, and sets of times. Moreover, the temporal expressions are not just full date and time expressions, as they may be underspecified, ambiguous, and anaphoric. Most state-of-the-art systems use thousands of hand-crafted rules, which probe words and their contexts in order to identify temporal expressions and assemble the information necessary to interpret them.

The Time Expression Recognition and Normalization (TERN[1]) evaluation effort, sponsored by the Automatic Content Extraction (ACE) program, covered both the identification and disambiguation subtasks. This evaluation campaign spanned through three different editions, in 2004, 2005 and 2007. Machine learning methods dominated the recognition task in TERN, achieving better performance than rule-based counterparts. However, for disambiguation, only rule-based were presented, attesting for the difficulty of developing machine learning methods for addressing the complete temporal reference resolution problem.

The TERN evaluation campaign relied on English annotated corpora that used the TIMEX2 scheme for marking the extent of temporal expressions and normalizing their values through the ISO-8601 format with some extensions. In brief, this scheme provides an inline `TIMEX2` tag for annotating temporal expressions in a textual document. There are six attributes defined for the `TIMEX2` tag and the values of these attributes express the semantics of a temporal expression:

- `VAL`: This attribute contains a normalized value for the date, time or duration covered by the annotated expression, in an format similar to that of ISO-8601. It can take one of three basic types of values:

[1] `http://timex2.mitre.org/tern.html`

  - Specific points in time are expressed as a string matching the general pattern **dddd-dd-ddTdd:dd:dd.d+**, which can be interpreted as a sequence of **year-month-dateThour:minute:seconds**. These strings may be truncated from the right, indicating points of coarser granularity. Any place in the strings can also be filled with a placeholder **X**, which indicates an unknown or vague value, and there are also 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 **dddd-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 an format similar to ISO-8601.
- `ANCHOR DIR`: This attribute captures the relative direction or 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 comment made by the annotator and it is ignored from the point of view of automated processing.

When using TIMEX2 annotations, the temporal reference recognition sub-task is concerned with finding the boundaries of temporal expressions in texts, while the temporal reference disambiguation task requires determining the values for the different TIMEX2 attributes.

A more recent standard, named TimeML[2], has been developed in the context of three AQUAINT workshops and projects. The first corpus produced with this annotation scheme was TimeBank, which is used in the TempEval-2[3] challenge. This annotation scheme uses a TIMEX3 tag to delimit temporal expressions in the text, which has slightly different attributes from the ones used in the TIMEX2 standard. In our work we used datasets encoded in TIMEX2 and TimeML, but we transformed the TimeML annotations into TIMEX2 so as to use a consistent working basis.

TempEx was the first TIMEX2 tagger, consisting of a relatively simple Perl script that implements part-of-speech tag heuristics using finite state automata, also performing limited disambiguation on the recognized expressions [7]. It handles both absolute times (e.g., *June 2, 2003*) and relative times (e.g., *Monday*) by means of tests on the local context of the references. Lexical triggers like *today*, *yesterday*, and *tomorrow*, when used in a specific sense, as well as expressions which indicate a positional offset, like *next month* or *this coming Thursday* are resolved based on computing direction and magnitude with respect to a reference time, which is usually the document publication time. Bare day or month names (e.g., *Thursday*, or *February*) are resolved based on the tense of neighboring past or future tense verbs, if any. Signals such as *since* and *until* are also used, along with information from nearby dates. TempEx was provided to all TERN participants, and it has since then been used as a popular baseline for measuring performance on new data.

[2] `http://www.timeml.org/`
[3] `http://www.timeml.org/tempeval2/`

Chronos[4] is a more advanced system that was also developed in the context of the TERN campaign [12]. Text processing in Chronos involves tokenization, statistical part-of-speech tagging, and multiword recognition based on a list of 5000 entries from WordNet. The core of the system consists of a set of approximately 1000 rules that recognize temporal constructions and gather information about them, which is latter used in the process of normalization. The system also uses a set of composition rules which resolve ambiguities when multiple annotations are possible. The disambiguation stage is based on a sequence of operations involving anchor selection for the case of anaphoric expressions, date normalization, and TIMEX2 attribute normalization. Although capable of achieving a reasonably good performance, Chronos involved a substantial effort in its development, and the system is not particularly generalizable.

Several machine learning TIMEX2 recognition systems have also been described in the literature [5,2,1]. For instance Kolomiyets and Moens compared two approaches for the automatic recognition of temporal expressions based on a Maximum Entropy supervised model, namely a token-based and a constituent-based approach [5]. The token-based approach used feature vectors comprising tokens in lowercase, part-of-speech tags, character type and character type patterns, together with the standard B-I-O encoding which subdivides token classifications as either being begin-of-reference (B), continuation-of-reference (I), or other-token-type (O). The constituent-based approach classified entire phrases as temporal expressions or not, restricting the classification to particular phrase types and grammatical categories. Feature vectors for this case included the head phrase, head word, a part-of-speech tag for the head word, and the character type and character type patterns for the head word and for the entire phrase. Results showed that both approaches have a good performance, with the constituent-based method slightly outperforming the token-based one.

For the disambiguation task, however, the usage of machine learning involves several hard challenges. There are a potentially unlimited number of classes (i.e., temporal values) and many expressions require either non-local context for interpretation (i.e., they are anaphoric or deictic) or a significant amount of temporal computation with respect to contextual information [1]. The work that is perhaps closest to ours is that of Ahn et al., where the authors presented a system[5] that uses machine learning together with a small set of hand-crafted, context-independent rules to achieve state-of-the-art disambiguation performance [2]. In our approach we used a more robust method for addressing the recognition stage (i.e., a CRF model instead of an HMM), and also a simpler method for performing the disambiguation (i.e., a regression model instead of a complex pipeline of classifiers). The following section details the proposed method.

3 The Proposed Approach

The proposed method for resolving temporal references is an instance of stacked learning, a machine learning paradigm that suggests constructing a combined model from several simpler learners [17]. 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.

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

Figure 1 provides an illustration for the general stacked learning procedure. A first level learner corresponds to a Conditional Random Fields (CRF) tagger for identifying temporal

[4] `http://qaserver.fbk.eu:8080/chronos/english`

[5] `http://ilps.science.uva.nl/Resources/timextag/`

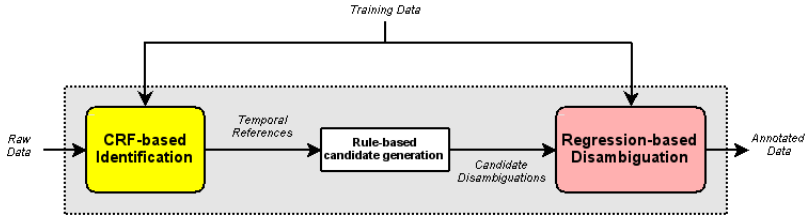

Fig. 1. Temporal reference resolution with stacked learning.

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 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 section details the learning approaches used in both levels.

3.1 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 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.

Thus, the tagging problem involves separate B, I and O 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 step.

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 sequence tagging model known as Conditional Random Fields offers an efficient and principled approach for addressing this problem [6].

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:

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

In the equation, ϕ are potential functions parametrized by Λ. Assuming ϕ_c factorizes a log-linear combination of arbitrary features computed over the subsequence c, then $\phi_c(y_c, x_c; \Lambda) = exp(\sum_k \lambda_k f_k(y_c, x_c))$ 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), thus 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 interdependencies. The list of features considered in our experiments includes context words (i.e., the token identity within a 1-token window of the target token), lexicons (i.e., whether the token appears in a list of bare weekday names, month names, or specific words associated with temporal expressions), regular expressions (e.g., 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) and prefix-suffix (i.e., the four letter prefix and suffix of the word).

3.2 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.

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 pattern, which may refer to a small lexicon of names, units, and numeric words, and a generator that produces values for the TIMEX2 VAL attribute. Rules are applied using regular expressions and their application involves 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 1 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 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 API[6], which allows us to easily manipulate time expressions. Point time expressions can be easily represented by intervals, if we consider the granularity of that time expression. If the granularity is, for instance, one month, we represent that by an interval that 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.

[6] 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`
Point	16	`(NUMERIC_DAYS | MONTHSPEC).? (of)? (YEAR)?`
		`DAYSPEC? ?Numeric31 (Numeric31|MONTHSPEC) YearFourDigitOrYearTwoDigits`
		`MONTHSPEC.? Numeric31(st|nd|rd|th)?`

Table 1. Rules for generating candidate disambiguations.

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 interval.
- The number of milliseconds from midnight UTC of January 1, 1970, to the end of the interval.
- The duration, in milliseconds, of the 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 the different temporal expressions occurring in the training data.

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 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 [5]. 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),

		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 2. The datasets used in our validation experiments.

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 [7]. In our experiments, we used Weka's implementation with a Radial Basis Function (RBF) as kernel.

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

4 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.

4.1 Gold-Standard Data and Quality Metrics

Our validation methodology is based on the standard NER evaluation setup, which involves comparing the annotations produced by an automated system against gold-standard annotations provided by human experts. As gold-standard annotated data, we used one English dataset with TIMEX2 annotations, namely WikiWars, and two English dataset with TimeML annotations, namely TimeBank 1.1 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 [14]. Section 2.2 briefly described the TIMEX2 annotation standard and Table 2 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 a news 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 correct references identified/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

	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**	0.92	**0.93**	0.82	**0.84**	0.83	0.75	0.62	0.68
Context words, rules and Pref_Suf	0.89	0.86	0.87	0.78	0.66	0.72	0.76	0.61	0.68
All features	0.90	**0.88**	0.89	0.81	0.81	0.81	0.76	**0.63**	0.68

Table 3. 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 4. The obtained recognition results for the different classes that were considered.

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.

4.2 The Obtained Results

We compared the proposed approach using different features for the CRF temporal expression identification model. Section 3.1 described the complete set of features that we considered and Table 3 presents the obtained results across the three different document collections. The results show that, the CRF tagger using a richer feature set achieves the best results, although Prefix and Suffix features seem to be the least informative. This can be due to the many different prefixes and suffixes temporal expressions can have, not corresponding to a specific pattern. In Table 4 we can see the results obtained by the CRF model that considered the full set of features, in all temporal reference chunk types.

The obtained results in the recognition sub-task are also comparable against those that have been previously reported over the same datasets – see Table 2.

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 points. In these experiments, we performed the recognition through the CRF model considering the entire set of features. Table 5 presents the obtained results, showing that, overall, the proposed approach is comparable or even outperforms other state-of-the-art systems. Although the values shown in Table 2 for the state-of-the-art disambiguation performance appear 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.

	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. The obtained results for the disambiguation sub-task.

The values of zero shown in Table 4, 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 2. 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 [9].
- 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 annotations.

5 Conclusions and Future Work

This paper presented a supervised machine learning method for resolving temporal references in text. The proposed method is an instance of stacked learning [17], 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.

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.

Resolving geospatial references over text also has important applications and equally challenging problems [8]. In the future, we plan to study integrated machine learning methods for resolving spatio-temporal references in text.

References

1. D. Ahn, S. F. Adafre, and M. de Rijke. Extracting temporal information from open domain text: A comparative exploration. *Digital Information Management*, 1(3), 2005.
2. D. Ahn, J. van Rantwijk, and M. de Rijke. 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*, 2007.
3. O. Alonso, M. Gertz, and R. Baeza-Yates. Clustering and exploring search results using timeline constructions. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, 2009.
4. R. Florian, A. Ittycheriah, H. Jing, and T. Zhang. Named entity recognition through classifier combination. In *Proceedings of the 7th Conference on Natural Language Learning*, 2003.
5. O. Kolomiyets and M.-F. Moens. Comparing two approaches for the recognition of temporal expressions. In *Proceedings of the 32nd Annual German Conference on Artificial Intelligence*, 2009.
6. J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning*, 2001.
7. I. Mani and G. Wilson. Robust temporal processing of news. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, 2000.
8. B. Martins, I. Anastácio, and P. Calado. A machine learning approach for resolving place references in text. *Proceedings of the 13th AGILE International Conference on Geographic Information Science*, 2010.
9. P. Mazur and R. Dale. A rule based approach to temporal expression tagging. In *Proceedings of the International Multiconference on Computer Science and Information Technology*, 2007.
10. A. McCallum and W. Li. 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*, 2003.
11. D. N. and S. Sekine. A survey of named entity recognition and classification. *Linguisticae Investigationes*, 1(30), 2007.
12. M. Negri and L. Marseglia. Recognition and normalization of time expressions: ITC-irst at TERN 2004. Technical Report WP3.7 – MEANING Project Deliverable, ITC-irst, Trento, 2004.
13. E. Sang and F. D. Meulder. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the 7th Conference on Natural Language Learning*, 2003.
14. E. Saquete. Terseo + t2t3 transducer: a systems for recognizing and normalizing timex3. *Proceedings of the 5th International Workshop on Semantic Evaluation*, 2010.
15. M. Verhagen and J. L. Moszkowicz. Temporal annotation and representation. *Language and Linguistics Compass*, 3(2), 2009.
16. I. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, 2000.
17. D. Wolpert. Stacked generalization. *Neural Networks*, 5(2), 1992.
18. G. Zhou and J. Su. Named entity recognition using an HMM-based chunk tagger. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, 2002.