TakeLab

Laboratorij za analizu teksta i inženjerstvo znanja
Text Analysis and Knowledge Engineering Lab
Sveučilište u Zagrebu, Fakultet elektrotehnike i računarstva
Unska 3, 10000 Zagreb, Hrvatska

UNIVERSITY OF ZAGREB
FACULTY OF ELECTRICAL ENGINEERING AND COMPUTING

BSc THESIS No. 3795

Temporal Expression Tagging for Croatian Texts

Luka Skukan

Zagreb, June 2014

UNIVERSITY OF ZAGREB
FACULTY OF ELECTRICAL ENGINEERING AND COMPUTING
BECHELOR THESIS COMMITTEE

Zagreb, June 11th, 2014

BACHELOR THESIS ASSIGNMENT No. 3795

Student: **Luka Skukan (0036465873)**
Study: Computing
Module: Software Engineering and Information Systems

Title: **Temporal Expression Tagging for Croatian Texts**

Description:

Temporal expression tagging refers to identifying the parts of text that denote intervals or moments in time and normalizing them to canonical forms. This is an important process in semantic text analysis and a prerequisite for more advanced information extraction methods, such as event extraction, document summarization, and temporal and causal reasoning. Previous research has shown that the task can be solved efficiently using methods based on a hand-crafted set of rules.
The topic of this thesis are the rule-based approaches for temporal expression tagging. Provide an overview of the various methods for detecting and normalizing temporal expressions, with a focus on rule-based methods, including the HeidelTime temporal expression tagger (Strötgen and Gertz, 2013). Develop and implement a temporal expression detection and normalization stystem for the Croatian language as an extension of the HeidelTime tagger. Compile a text corpus manually annotated with temporal expressions, along the lines of the WikiWars corpus (Mazur and Dale, 2010). Perform an experimental evaluation and a detailed error analysis. All references must be cited, and all source code, documentation, executables, and datasets must be provided with the thesis.

Thesis submitting date: June 13th, 2014

Mentor:

Prof. Jan Šnajder, PhD

Committee Chair:

Prof. Krešimir Fertalj, PhD

Committee Secretary:

Prof. Ivica Botički, PhD

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA
ODBOR ZA ZAVRŠNI RAD MODULA

Zagreb, 11. lipnja 2014.

ZAVRŠNI ZADATAK br. 3795

Pristupnik: **Luka Skukan (0036465873)**
Studij: Računarstvo
Modul: Programsko inženjerstvo i informacijski sustavi

Zadatak: **Označavanje vremenskih izraza u tekstovima na hrvatskome jeziku**

Opis zadatka:

Označavanje vremenskih izraza podrazumijeva prepoznavanje izraza u tekstu koji se odnose na vremenske trenutke ili intervale te normalizaciju tih izraza na kanonske oblike. Ovaj je postupak važan za semantičku analizu teksta te je preduvjet za napredne postupke ekstrakcije informacija kao što su ekstrakcija događaja, sažimaje dokumenata ili rasuđivanje o vremenskim i uzročnim odnosima. Istraživanja su pokazala da se ovaj zadatak može učinkovito riješiti postupcima temeljenima na ručno izgrađenim pravilima.

U okviru završnoga rada potrebno je proučiti postupke za prepoznavanje i normalizaciju vremenskih izraza, s naglaskom na postupke temeljene na pravilima, uključivo sustav HeidelTime (Strötgen i Gertz, 2013). Razraditi i implementirati postupak prepoznavanja i normalizacije vremenskih izraza u tekstovima na hrvatskome jeziku kao nadogradnju sustava HeidelTime. Izgraditi ručno označenu zbirku vremenskih izraza u tekstovima na hrvatskome jeziku po uzoru na zbirku WikiWars (Mazur i Dale, 2010). Radu priložiti izvorni i izvršni kod razvijenog sustava, skupove podataka i programsku dokumentaciju te citirati korištenu literaturu.

Zadatak uručen pristupniku: 14. ožujka 2014.
Rok za predaju rada: 13. lipnja 2014.

Mentor:

Doc. dr.sc. Jan Šnajder

Djelovođa:

Doc. dr.sc. Ivica Botički

Predsjednik odbora za završni rad modula:

Prof. dr.sc. Krešimir Fertalj

ACKNOWLEDGEMENTS

I would like to take the time to thank my supervisor, doc. dr. sc Jan Šnajder, for giving me both assistance and a degree of independence. Many thanks, likewise, to dr. sc. Goran Glavaš for helping me in the later stages of this thesis' creation.

I would also like to thank my cousin, Borna Skukan, without whose help the corpus might have never been fully annotated. Thank you for finding more enthusiasm for manual annotation than I ever could.

Finally, thanks to Juho Hankala, Ville Vartiainen and my mother, Ivanka, for going through the thesis and giving me pointers and corrections where necessary. Without them, this thesis would be an incomprehensible mess.

CONTENTS

LIST OF TABLES

1. Introduction

The concept of time is one of the core concepts of reality itself. We think and live in terms of moments in time, periods, durations and intervals. We understand complex events and their interconnection due to our comprehension of the notion of time. Because of our innate understanding and need for the concept, it permeates our languages. While it might often be trivial for a human, understanding time is difficult for a computer.

The human need for computers to understand natural languages gave birth to a new area of artificial intelligence – natural language processing (NLP). It is a field of computer science and linguistics which deals with interaction between computers and humans using natural (human) languages. Its primary goals are "understanding" natural language input - deriving its meaning in a form usable by computers - and generating natural language output. One of its tasks is temporal expression tagging, which deals with the identification of parts of input which denote time, such as moments and intervals, and normalising them to canonical forms. It is an important process in semantic text analysis and a prerequisite for more advanced information extraction methods, such as event extraction, temporal and causal reasoning and document summarisation. Temporal expression extraction usually consists of two steps:

1. **Temporal expression identification** Consists of locating temporal expression, determining their span and tagging them.

2. **Temporal expression normalisation** Describing the temporal value of the expression in one of the canonical forms for temporal expressions.

This thesis addresses the task of extraction on a textual corpora written in the Croatian language. The two methods that are used for temporal expression extraction, as well as in most of natural language processing, are the rule-based approach and the statistical approach. This thesis focuses on the rule-based approach and uses the HeidelTime engine for the task.

The data set used in the thesis is a corpus of the Croatian Wikipedia articles about famous wars, inspired by the WikiWars corpus. The corpus was tagged and split in half, with one half used as a development set for writing the rules and the other for testing the accuracy of the written rules. The corpus was named WikiWarsHR.

Chapter two offers a summary of the history of temporal expression tagging and related work, as well as a comparison between the approach used in this thesis and the approaches listed. In chapter three, the concepts and mechanism of the HeidelTime engine are described in detail. Chapter four deals with the extension of the HeidelTime engine with support of the Croatian language. In the fifth chapter, the creation of the WikiWarsHR corpus of tagged articles is discussed. The penultimate chapter lists and discusses the results of experimental evaluation performed on the corpus. The last chapter offers a conclusion and lists possible future improvements.

2. Related work and summary

2.1. Summary and history

Temporal expression extraction's importance lies in the fact that it is usable in higher-level analysis, such as event extraction, automated translation, text summarisation, question answering, as well as other processes which rely on temporal orientation for results. It is generally split into two discrete parts – temporal expression identification and normalisation. Identification refers to locating temporal expressions and their extent in text, while normalisation refers to placing these events on points or areas on a time-line, be it explicit or relative to another point. Other than explicit expressions and expressions relative to another expression – the current focus of the text or the document creation time – the expressions can also denote duration in time or a set of events, which denote periodic occurrence of an event. Examples for each are given in Table 2.1. While this sort of temporal expression classification is not formal, it is recognised and used in several temporal expression normalisation standards, such as TIMEX2 and TimeML(Pustejovsky et al., 2003a).

Table 2.1: Examples of temporal expression types in Croatian

Type	Example
Explicit	16.06.2014.
Relative to Focus	Godinu dana kasnije
Relative to Document Creation Time	Danas
Duration	pet sati
Set	Petkom

Temporal expression extraction is a relatively new addition to the field of natural language processing. It was first mentioned in the MUC-7 competition, in 1998, which was sponsored by DARPA. The following year, the ACE Program (*Automatic Content Extraction*) was started. It outlines its goal as automatic entity, relation and event ex-

traction. While the program also deals with extraction from image and audio sources, its primary focus was always text, specifically in English, Arabic and Chinese. While temporal expression extraction was already implicitly present in some of these tasks, it was explicitly introduced in 2004, in the form of a new ACE task: TERN (*Time Expression Recognition and Normalization*). Since then, many, if not most, innovations in the area of temporal expression extraction come from TERN.

The two approaches usually employed for temporal expression extraction are the rule-based approach, which employs pattern-matching against hand-written rules, and the machine learning approach. In the early days of temporal expression extraction, the rule-based approach was the dominant one. One such example is the Chronos system (Negri and Marseglia, 2004). These systems generally recorded very good results, but developing them took more time since the rules have to be written by hand. For that reason, the machine learning approach started to be more commonly used. They likewise showed good results, but were easier to implement. Examples include (Kolomiyets and Moens, 2009) and (UzZaman and Allen, 2010). However, recently there have been some new rule-based systems, such as HeidelTime (Strötgen and Gertz, 2010) and SUTime (Chang and Manning, 2012), which have achieved top results in TempEval, a part of the SemEval[1] competition focusing on temporal tasks.

Normalisation is still mostly rule-based. Attempts at a viable machine learning approach to expression normalisation have been made, but have shown to be insufficient at the task thus far (Ahn et al., 2005). In a rule-based approach, like the one used in this thesis, these rules are written together with the identification rules. While this approach requires more work on hand-writing rules, its results are much better. Another pervasive issue with normalisation, and historically the source of most normalisation errors, is keeping track of focus for determining values of relative expressions. Modern rule-based systems usually keep track of both focus relative to document creation time and current focus of the text, thus lowering the number of such errors.

Normalisation must reduce an expression to a canonical form. Many standards exist, but the most popular ones include TIMEX2, TimeML and TIDES standards. The first two were developed for specific corpora, but have since been used elsewhere, while the TIDES standard was developed as part of DARPA's **T**ranslingual **I**nformation **D**etection, **E**xtraction, and **S**ummarization research program, from which it takes its name (Ferro et al., 2005a). The TIMEX2 format was developed under TIDES sponsorship, primarily for the ACE TERN 2007 task corpus, which consists mostly of

[1] Semantic Evaluation – SemEval webpage for 2013: http://www.cs.york.ac.uk/semeval-2013/

news articles and news and conversation transcripts (Ferro et al., 2005b). TimeML was developed as a markup language for the TimeBank news article corpus (Pustejovsky et al., 2003b) during the TERQUAS (*Time and Event Recognition for Question Answering Systems*) workshop (Pustejovsky et al., 2003a). TimeML features a subset, TIMEX3, which has since gained popularity. In March 2009, its variant, ISO-TimeML, has become an ISO standard for temporal expression annotation.

The mentioned corpora, and many corpora besides, are primarily based on news articles. However, other types of corpora have emerges, such as the WikiWars corpus, which is based on historical Wikipedia articles (Mazur and Dale, 2010). These articles are interesting and useful to the field due to their narrative style and the fact that they are rich with temporal expressions. Versions of the WikiWars corpus have been made for other languages, such as the WikiWarsDE corpus (Strötgen and Gertz, 2011) for German and the WikiWarsVN corpus (Strötgen et al., 2014). Such corpora, as they are based on narrative historical articles, are usually very structured and good results can be achieved with focus management.

Due to the fact that the output of one natural language tagger is often the input of another – for example temporal expression can be used in event tagging – natural language processing pipelines are a natural development. The best known amongst them is Apache's UIMA pipeline, an implementation of the UIMA specification (Ferrucci and Lally, 2004). Such a pipeline enables users to easily decompose processing into autonomous application components and add and remove components freely while communicating through the pipeline via XML descriptor files. Some annotators, like HeidelTime, are written to fit into such document processing pipelines.

2.2. Related work

2.2.1. Temporal Expression Extraction

The work of Strötgen at al. (2010) describes the HeidelTime Engine, as created for the TempEval-2 challenge. It was written in Java and uses hand-crafted rules, with support for rule sets in different languages. The TempEval-2 challenge featured temporal tagging, event and temporal relation extraction into the TimeML format – Tasks A, B and C. HeidelTime participated only in task A. They chose to implement a rule-based approach for both identification and normalisation, as to allow for modular extensions and increase control over the normalisation. The HeidelTime tagger was specifically implemented to be a part of a document-processing pipeline. Two sets of rules were

used for the evaluation, one optimised for precision, the other for recall[2].

They entered the competition with a relatively small set of rules, 43 rules in the precision set and 45 in the recall set. However, they achieved top results in identification for task A: (P – 90%, 80%; R – 82%, 91%; F1 – 86%, 86%) for the precision and recall sets, respectively. The precision-optimised set also achieved the best results in type and value normalisation, with scores of 96% for type and 85% for value. However, both were evaluated even for only partial matches, thus raising the scores. This showed that rule-based systems are still a viable option and outlined a few advantages of theirs – easier and more precise normalisation, as well as easy incorporation of additional rules.

The system was later improved and reused in TempEval-3, on English and Spanish corpora (Gertz, 2013). Similarly good results were achieved for identification, (P – 93%; 96%; R – 87%, 85%; F1 – 90%, 90%; Type – 82%, 87%; Value – 77%, 85%) for English and Spanish respectively, using relaxed matching rules. Rules have also been implemented for Vietnamese, Arabic, French and Italian, with similar results (Strötgen et al., 2014), as well as Russian.

Chang et al. (2012) present SUTime, a rule-based temporal tagger built on regular expression patterns, much like HeidelTime, using finite automata for parsing. Again, like HeidelTime, SUTime is implemented in Java to identify and normalise temporal expressions and output them as TIMEX3-annotated text. There are other similarities. SUTime is one of the systems inspired by HeidelTime's TempEval-2 performance. Its results, compared to HeidelTime's TempEval-2 performance, are very close. It achieves identification scores of (P – 88%; R – 96%; F1 – 92%; Type – 95%; Value – 68%). Its results for recall and F1 are slightly larger than HeidelTime's, while its precision is slightly lower and its value normalisation is significantly worse.

However, if strict evaluation is applied to value and type scoring, SUTime achieves closer results to HeidelTime – 89% type scoring, compared to HeidelTime's 88% and 74% value scoring, compared to HeidelTime's 76%.

2.2.2. Tagged Corpora

The TimeBank corpus is presented in a paper by Pustejovsky et al. (2003). The creation of the TimeBank corpus is linked with the development of the TimeML language specification. Its purpose was twofold – to create a gold standard human-annotated

[2]See Chapter 6 for explanation of precision and recall

corpus of temporal expressions, events and temporal relations and to create a better annotation markup specification. The corpus consist of 183 annotated news articles, with over 61,000 tokens and and 27,592 tags. These tags include both temporal expressions and more complicated kinds of relations. The corpus was widely used and accepted as a gold standard. It was later revised to be more consistent and robust (Boguraev et al., 2007).

The work of Mazur and Dale (2010) describes the WikiWars corpus. It was created to correct what was considered a fatal flaw amongst other available corpora for English – the briefness of articles and low complexity of temporal discourse. It consists of 22 articles from the English Wikipedia, on the topic of wars, annotated with TIMEX2. The type of these documents, historical narrative, provides a measure of length and a wealth of temporal expressions. The final corpus is much larger than the TimeBank corpus, resulting in 120,000 tokens and 2,671 temporal expressions. The tagging was performed both automatically and by hand, to avoid "annotator blindness". While the density of tags is lower overall, the TIMEX2 standard is used only to annotate temporal expressions, not events or temporal relations, so such a result is to be expected.

The corpus provided the community with a new, sizeable tagged corpus, different both in structure and type of text. It was acclaimed enough to inspire at least two versions in other languages – WikiWarsDE for German (Strötgen and Gertz, 2011) and WikiWarsVN for Vietnamese (Strötgen et al., 2014).

Inspired by the WikiWars corpus for English, a German version was developed and unveiled in a paper by Strötgen and Gertz (2011). The reasons for creating a German version were largely the same as for the original – narratives offer many temporal expressions, lengthy articles and many relative temporal expressions. This corpus became the first publicly available temporal annotated corpus for German. The WikiWars annotation guidelines were followed and the corpus also consists of 22 documents detailing the courses of the same wars as the English version, tagged partially automatically (using HeidelTime), and partially by hand. The corpus is smaller than the original WikiWars, consisting of 95,604 tokens and 2,240 temporal expressions.

2.3. Comparison with this thesis' approach

HeidelTime and SUTime both have very similar approaches to temporal expression tagging. Both use regular expression rules, track focus and have TIMEX3 output. As

this thesis focuses on extending HeidelTime for Croatian, its approach is nearly identical to the original one. However, as Croatian is a more inflected language than English and HeidelTime rules are defined over tokens, not lemmas, the rule set for Croatian is much larger than the one for English (199 compared to the original 43/45). For the same reason, custom token and sentence splitting and part-of-speech tagging had to be used. Much of the functionality remains embedded in the language-independent HeidelTime engine and was not modified. The engine's modularity was used to improve processing of historical texts.

The WikiWarsHR corpus is similar in structure to the WikiWars and WikiWarsDE corpora. It also consists of 22 articles, 21 of which are on the same topics as the English and German versions. However, the texts are in Croatian and are annotated with TIMEX3 instead of TIMEX2. This allow the tags to be expanded to cover more than just temporal expressions in the future, using the full TimeML set of tags. The corpus is also smaller, roughly half the size of the WikiWars English corpus and $\frac{2}{3}$ the size of the German corpus.

WikiWarsHR is similar to the TimeBank corpus in its annotation language of choice. However, while TimeBank makes full use of TimeML, this corpus' annotations are limited solely to temporal expressions.

3. HeidelTime implementation details

As the tagger for the Croatian language is an extension of the HeidelTime model, it shares its basic concepts. HeidelTime is written in Java and relies heavily on the object-oriented abstractions it offers. It exists in two separate versions – a basic HeidelTime engine which takes pre-processed text as input and is meant to be used as a component in the UIMA[1] (Unstructured Information Management Architecture) pipeline, as well as a standalone version.

The HeidelTime framework also supports registration of post-processing modules. They can be used to perform adjustments on the normalised values of tagged tokens, or execute functions post-evaluation.

To simplify adding markup and data to tokens and sentences, a common data structure is used – UIMA pipeline's Common Analysis Structure (CAS), as implemented in the Java programming language. At the start of the process, the CAS contains only the text itself. Each processing step can then add further data to the CAS object. The CAS object is especially suited for usage in a larger NLP pipeline – for example, temporal expression tagging could follow named entity extraction and precede event tagging.

3.1. Pre-processing

The standalone version, which was used in this thesis, takes raw text as input and does the pre-processing itself before performing the temporal expression tagging. The pre-processing consists of three steps:

1. Splitting the input text into sentences;
2. Splitting the input sentences into tokens (words, punctuation, etc.);
3. Adding POS (part-of-speech) tags to tokens – Determines the type of the word, tense, gender, etc.

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

These steps can be completely separate if different modules are used for pre-processing, but multiple steps can be connected into one. The order of token and sentence tagging is not strictly relevant, as sentence boundaries can be determined either by identifying stop tokens[2], or as a separate step. The algorithm is described, in broad terms, in Algorithm 1. As noted, due to the possible variations in token and sentence tagging, this is not the only possible version of the algorithm and depends on the specific implementation.

Algorithm 1 Pre-processing Algorithm

```
sentences ← sentenceSplit(input)
for sentence in sentences do
    tokens ← tokenSplit(sentence)
    setTokens(sentence, tokens)
end for
for sentence in sentences do
    tokens ← tokens in sentence
    for token in tokens do
        addPOSTags(token)
    end for
end for
```

Each of the steps in the algorithm adds data to the CAS object, in relation to the currently processed object. Sentence splitting denotes the indices of start- and end-points of sentences, while token splitting does the same for tokens. The POS tagging process adds part-of-speech data to each token in turn. The pre-processed text is then forwarded, with data added to the CAS, to the processing engine.

3.2. Temporal expression identification

If we describe temporal expressions as a list of three-tuples $te_i = (e_i, t_i, v_i)$, where e_i is the temporal expression, t_i its type and v_i its value, the extraction step provides normalised values t_i and v_i for given e_i. The format of the normalised values adheres to the TimeML standard[3], or, more precisely, its subset TIMEX3. As such, the type of an expression (t_i) can be *Date*, *Time*, *Duration* or *Set*. The value (v_i) property specifies the value of the temporal expression and partially depends on the type. Both type and date

[2]Tokens indicating the end of a sentence – examples include !, ?, ., etc.

[3]http://www.timeml.org/site/publications/timeMLdocs/timeml_1.2.1.html

use an extended version of the ISO 8601 norm to describe their normalised values, as specified by TimeML. The *Date* type describes an expression signifying a date without a specific time assigned. *Time* is more specific than *Date* – it has a date value, but a time value as well. The value can be very specific, such as 2013-05-15**T**12:14:00[4], or more vague, such as 2013-05-15**T**MO[5]. Expressions of type *Duration* mark something that lasts for a certain period of time, such as "one year" or "twenty-five minutes". They are expressed as "P$n_i u_i$", where n_i is a number and u_i is the symbol for a unit. The expression "one year" would normalise to "P1Y". *Set* describes a set of times, such as "every October" or "daily". Some examples for normalisation of temporal expressions in Croatian are given in Table 3.1. They assume the current date is May the 16th 2014.

Table 3.1: Normalisation examples

Expression	Type	Value
12.10.1995.	DATE	1995-10-12
Zima 1945.	DATE	1945-WI
Danas u 17 sati	TIME	2014-05-16T17
Jutros	TIME	2014-05-16TMO
dva sata	DURATION	PT2H
godina dana	DURATION	P1Y
svakih tjedan dana	SET	P1W
svake veljače	SET	XXXX-02

In case of ambiguity, HeidelTime can use POS tags to attempt to infer the value. For example, in the expression "in May, I went to Peru". Assuming the current context is April 2014, it could refer to May 2014 or May 2013. However, the past tense of the verb "to go" indicates it happened in the past and the year 2013 is inferred even though the current context year is 2014.

In addition to the value itself, a TIMEX3 expression can also have a modifier which specifies the value further – e.g. "late in the summer of '69" would, assuming the context is the 20th century, have the value "1969-SU" as well as the modifier "END". The modifiers are listed in the TIMEX3 specification. TIMEX3 and TimeML permit other attributes for expanded verbosity, but they are not used in the scope of this thesis and will not be discussed.

[4]The expression on the left-hand side of the uppercase T is the date, as YYYY-MM-DD, while the expression on the right-hand side of it is the time, in the format hh:mm:ss. Values other than years, months, days, hours, minutes and seconds are permitted by the norm.

[5]MO – The morning

In the matching and normalisation processes, HeidelTime uses three types of resources:

- Expression resources
- Normalisation resources
- Rule resources

Expression resources are files containing a list of regular expressions which describe a class of expression. One such resource might be *reMonthLong* which would contain (for the Croatian language) expressions "([Ss]iječanj | [Vv]eljača | . . .)". Normalisation resources contain rules or functions for normalising matched expressions into a canonical form prescribed by TimeML. For example, *normMonth*("Travanj") = "04". The rules employ these atomic resources, combined with other regular expressions, into complete identify-and-normalise expressions. Rules can also employ other constraints and directives, such as POS constraints, which require the expression or a certain part of an expression to match it, or offset rules, which tag and normalise only a certain part of the matched expression.

Table 3.2: Example of using resources for extraction

Expression resources	*rePartWords*("Početak | Sredina | . . . "), *reMonth*("Siječanj | Veljača | . . . "), *reYear*("[12]\d\d\d")
Normalisation resources	*normPartWords*("Početak") = "START", *normPartWords*("Sredina") = "MID", *normMonth*("Siječanj") = "01", . . .
Rule resource	date_r1 = pattern = "$(rePartWords)_{g1}$ $(reMonth)_{g2}$ $(reYear)_{g3}$.", value = "g3-*normMonth*(g2)", mod = "*normPartWords*(g1)"

In addition the these *positive* rules, which are used for processing valid matches, HeidelTime also permits *negative* rules, which are used to discard specific invalid matches. These can be useful if a rule matches a broad class of rules, but has specific cases where an invalid match would occur. One example of that, in English, would be the expression "ten years old". HeidelTime, using the resources for the English language, would recognise "ten years" as a ten year period (*P10Y*), but a negative rule would recognise that it is a part of a broader expression describing age, not a period

of time, and would discard it. An example of application of a positive rule using these resources if given in Table 3.2, using a simplified notation for improved readability.

The rules can then be applied to text. Taking the rule date_r1 from Table 3.2 and applying it to the expression "Sredina srpnja 2003." (the middle of July 2003) would yield the the result (type = "DATE", value = "2003-07", mod = "MID").

One may notice that the rule specified in Table 3.2 does not specify the type of the expression – the only mention of it being a date is in the name of the rule. HeidelTime handles the type distinction by splitting rule resources into separate resource files, depending on type, as well as using such rule conventions. As such, all rules pertaining to dates are in the *daterules* resources file and follow the naming convention *date_r.*....

Algorithm 2 illustrates how the annotator applies these rules. Each rule set is applied in order to each sentence. Then, two steps are executed to disambiguate under-specified values and to remove invalid temporal expressions. These two steps belong to the post-processing step and will be detailed there.

Algorithm 2 Rule Application

```
for sentence in document do
    addDatesToCAS(daterules, CAS)
    addTimesToCAS(timerules, CAS)
    addDurationsToCAS(durationruels, CAS)
    addSetsToCAS(setrules, CAS)
end for
for timex3 in CAS do
    disambiguateValues(CAS)
end for
for module in modules do
    for timex3 in CAS do
        process(module, timex3)
    end for
end for
removeInvalidFromCAS(CAS);
```

Under-specified values are HeidelTime's implementation of focus. For some expressions, it is trivial to determine their value – "May of 1802" is one such example. However, some rules have a value relative to some other token they refer to. We cannot know the value of the expression "a year earlier" if we do not know which year we are currently referring to. HeidelTime therefore normalises that value as an undefined

or relative value – in this case as "UNDEF-REF-last-year". These values can then be resolved later, using a reference time. This mechanism is also used for determining decades[6], holidays and other values which depend on focus for normalisation.

Finally, the HeidelTime processing engine allows the rules to use a handful of predefined functions during the normalisation phase. These functions are *SUBSTRING*, *UPPERCASE*, *LOWERCASE* and *SUM*. One can use these rules for simple text processing which does not require registration of further post-processing modules. For example, we can calculate the expression "19th century" by taking the number *19* and applying the *SUM* function on it as %SUM%$(n_i, -1)$ and getting $19 - 1 = 18$, which, in TIMEX3, would indeed mark the 19th century, or year *18??*[7].

3.3. Post-processing

HeidelTime executes post-processing in three separate steps – undefined value disambiguation, post-processing module execution and invalid annotation removal. The first and last step are required, while the second is optional - HeidelTime doesn't have to have any modules registered, but they add functionality to the post-processing engine.

Ambiguous values are under-specified. These values start with "UNDEF-REF-" or simply "UNDEF-". Such values are disambiguated using a reference time as a focus. The focus depends on the prefix. UNDEF- values are resolved using the document creation time (DCT) while the UNDEF-REF- values are disambiguated using a previously referenced date. Assuming the dct is 2014-05-16, the expression "tomorrow" would have the value "UNDEF-next-day". Such a value would be resolved as "2014-05-17", using the dct as focus. In case of a UNDEF-REF- value, all preceding extracted TIMEX3 of type *Date* are checked. The value of the closest previous date is then used for disambiguation. While this does not always yield correct results, as will be shown during evaluation in Chapter 6, it successfully disambiguates in the majority of cases.

The HeidelTime model allows post-processing modules, written as Java classes, to register themselves for execution. If any such modules are registered, they are sequentially executed on the tokens contained in the CAS. HeidelTime uses a post-processor for processing lunar holidays like Easter. It would be much too complicated to calculate the date using simple inline functions in the normalisation rule, so instead a special token is left in place of the value as a marker to the post-processor. The holiday

[6]Not originally – the implementation for Croatian offers a decade processor. See Section 4.3

[7]This way of marking centuries ignores the fact that year 1800 belongs to the 18th century, while year 1900 also belongs to the 19th

processor, for example, leaves a function with its parameters, an event and a relative offset. An example is given in Table 3.3, assuming the year is 2014. The token can, of course, take a different form, as long as it is distinct from valid TimeML values.

Table 3.3: Using a post-processing module

Step	Form
Identification	Uskrs ove godine
Normalisation	UNDEF-this-year-00-00 funcDateCalc(EasterSunday(YEAR,0))
Disambiguation	2014-00-00 funcDateCalc(EasterSunday(YEAR,0))
Post-Processing	2014-04-20

The invalid annotation removal step of post-processing resolves cases where annotations would overlap. We have two cases where this might happen:

- One annotation is contained within another annotation;
- Two annotations partially overlap over one or more elements.

A simple example of the first case would occur if we take the expression “June of 1985”. HeidelTime would recognise three expressions: “June”, “1985” and “June of 1985”. However, the first two expressions are subsumed in the third one and ought to be eliminated. HeidelTime’s invalid annotation removal will take care of such cases and, as long as all terms are contained within the span of one, most specific element, take the element with the largest extent.

The second case is a bit more complex, both in occurrence and resolution. It occurs on partial overlaps and is an consequence of incomplete rule resources. We might encounter it while processing the term “the middle of June, 1944". A set of poorly written rules might recognise it as two partially overlapping temporal expressions – “the middle of June” and “June, 1944”. This can be avoided with a a rule which would cover the entire expression, but happens due to human error, as the rules are hand-written. When this occurs, HeidelTime strives to maximise granularity. It assumes that a longer expression means a better granularity and takes the longest of the overlapping expressions.

4. Implementation for Croatian

The process of adjusting HeidelTime to work for Croatian texts consisted of two necessary steps: Writing wrappers for pre-processing software and crafting the expression, normalisation and rule resources. In addition to those steps, a post-processing module was written to bolster performance on historical texts. Each of these steps is detailed below.

4.1. Pre-processing – HunPos and CSTLemma

As mentioned in Chapter 3, Section 3.1, HeidelTime needs its text pre-processed - split into sentences and tokens and POS-tagged. The standalone version of HeidelTime does this during its pre-processing step, but does not know how to do it on its own. Instead, it invokes other software which does the task for it, using a wrapper written in Java. For pre-processing Croatian texts, CSTLemma (Jongejan and Haltrup, 2005) was used for sentence and token splitting, while HunPos (Halácsy et al., 2007) was used for POS tagging. Both these tools were not originally written to work with Croatian texts, but have been trained to work with them (Agić et al., 2013).

A wrapper class was written in Java to extend these tools' functionality to HeidelTime. The wrapper adheres to HeidelTime's standard wrapper interface, as found in its source code, and performs three functions key to Standalone HeidelTime's functionality:

- Splitting text into tokens by invoking CSTLemma;
- Adding POS tags and performing sentence splitting by invoking the HunPos tagger and translating them into a form recognised by HeidelTime;
- Correcting errors in sentence and token recognition.

The CST lemmatiser is capable of token splitting as well as token lemmatisation, but HeidelTime works with raw tokens, so the lemmas were discarded and only the lemmatiser's token splitting capabilities were employed. The Java wrapper simply

invokes the lemmatiser, written in C++, as a separate process and handles its output and input.

The HunPos tagger wrapper's role was two-fold. HunPos is invoked on the tokens produced by the lemmatiser and assigns POS tokens to them. However, the POS tags marked by the tagger also include tags for stop tokens - punctuation which marks the end of a sentence. After the wrapper is given these tokens, it can use them to perform sentence splitting. HunPos' tags, however, are not compatible with HeidelTime and need to be translated into TreeTagger's tags to be of any use in processing. Due to only partial compatibility, a simple scheme is used – only a handful of POS tag types are translated into meaningful POS tags, while the rest are marked as foreign words. The tokens which are translated are those referring to nouns and verbs and their tenses, as well as adjectives and adverbs. This allows us to handle most cases of ambiguity, while keeping the mapping simple.

These two steps would be enough to make the text parseable. However, they tend to, on occasion, produce incorrect results for token and sentence boundaries. Brackets are considered parts of the words they wrap around, while some punctuation marks are erroneously considered the end of a sentence. However, the most harmful instance of error during pre-processing is that HunPos recognises every dot as an end of a sentence. This is harmful because numeric representations of Croatian dates are delimited by dots – "The sixth of June 2013" would be "06.06.2013.". As was mentioned previously (Chapter 3, Section 3.2, HeidelTime's rules are applied to sentences. It will not recognise an expression as a match to any rule if it is split into two sentences in the CAS object. Therefore, it is necessary to correct these mistakes.

The wrapper's correction process consists of two sub-processes – token correction and sentence correction. Token correction is the simpler step; it splits tokens which begin in an opening bracket or end in a closing bracket into separate tokens. The most common example encountered was a bracket preceding a year, such as "(1948.". Since this is considered a single token by CSTLemma, it would not be parsed as a valid year by HeidelTime. Therefore, the token correction sub-process splits this into two tokens containing "(" and "1948.".

Sentence correction deals with an opposite problem - something that is a single sentence is often parsed as multiple sentences. The two most common cases of this occur on dots delimiting dates, as was mentioned in the previous paragraph, and on dashes markings spans of years (e.g. "1940. – 1952."). The correction method performs these repairs in two steps – correcting wrong sentence splits not containing dots and correcting sentence splits containing dots. These are separate because performing

the first is always valid and much simpler, while only some dots mark full stops. A simple and approximate approach was taken here, but was shown to work well in practice – if the dot is preceded by a number and followed by anything but an uppercase word, the two sentences are fused together. If it is preceded by a number and followed by an uppercase word which is a name of a month, they are likewise fused. Otherwise, the two sentences remain separate.

4.2. Processing Resources

The most critical part of extending HeidelTime to work for the Croatian language is implementing the rule resource set for Croatian. Words and terms are matched to these rules during expression identification and normalisation, as was described in Chapter 3. These resources were split, as is customary for HeidelTime, into three sets – the expression, normalisation and rule resources.

The expression resources, as mentioned previously, contain regular expressions containing identification rules for groups of terms. Some of the examples are *reMonthLong* and *reUnit* – words describing long month names (Siječanj – January, Veljača – February, etc.) and temporal units (day, week, month, decade, ...). While the number of rule groups is similar to those in the English and German resources, the number of rules is larger. This is mainly due to the fact that Croatian is a heavily inflected language. Since HeidelTime processes raw tokens instead of their lemmas, patterns had to be written for every form of of the word which might appear in texts. Experience has shown this to be a particularly large issue with Croatian nouns, due to declension, which is not present in English and far simpler in German. Implementing a lemma-based approach is possible, but was deemed to complex to be implemented in this thesis.

The normalisation resources, which map matched patterns to normalised values, follow a similar pattern as the expression resources. As every token matched via expression rules must be normalised, these resources likewise must contain all inflections of the word which might appear in text, paired with a normalised value. As such, the various forms of the Croatian word for February – “Veljača”, “veljača”, “Veljaču”,“ Veljači”, as well as many others – all map to the same normalised value, “02”. The job of writing these rules contains a lot of expression resource duplication and was, in this case, partially automated.

The implementation of rule resources consisted of combining the previous two sets into consistent identify–and–normalise rules, as described in Chapter 3. The rules

follow the form outlined in the same chapter and do not differ much from the rules for other languages in form, only in language-specific content. One notable exception are the decade-based rules, which originally always use the 20th century as their focus. As such, the English expression “the nineties” would always evaluate to “199”, ignoring the current focus. Due to the historical nature of the corpus used in this thesis, this was changed to work with the current century focus by use of post-processing module, as detailed in the next section.

All these resources, with the exception of a handful of automatically generated normalisation rules, were hand-written. The process of creating them consisted of two steps:

– Translating and adjusting English and German resources;

– Adding and improving rules on a development set of tagged texts.

While the second step would be sufficient for the corpus used in this thesis (see Chapter 5), the first step adds robustness to the rules. Developing them solely on a single domain of texts (historical narratives, in this case) would likely result in poorer results in other domains. As the English and German resources were developed on a broader set of corpora, some of their versatility is borrowed by the Croatian resource set.

Table 4.1: Examples of resources for Croatian

Expression resources	
Description	Resource
4-digit year	[12]\d\d\d
Part of day	([Uu]jl[Jj])utr[oa], [Pp]o(po)?dne, [Pp]oslijepodne, etc.
Approximate	[Nn]ajmanje, [Bb]arem, [Vv]iše od, etc.
Normalisation resources	
Description	Resource
Months	"[Ss]i?ječ(anj|nj[au])","01", "[Vv]eljač[aeiou]","02", etc.
Weekdays	"[Pp]onedjeljak","1", "[Uu]torak","2", etc.
Rule resources	
Description	Resource
Year	EXTRACTION="%reYear4Digit",NORM_VALUE="group(1)"
Month	EXTRACTION="(%reMonthLong|%reMonthShort)", NORM_VALUE="UNDEF-year-%normMonth(group(1))"

The final result is a set of 199 rule resources: 123 for dates, 37 for time, 24 for durations and 15 for sets. This is significantly larger than the number of rules for English or German. The number could likely be somewhat reduced if several rules were combined into a more general one, but would still remain larger than the number of rules for English or German. This is partially due to the more complex nature of the Croatian language and partially due to some specific cases encountered while refining the rules over a development set of annotated text. Some examples of more important resources are given in Table 4.1.

4.3. Post-processing extensions

HeidelTime's post-processing is designed to be as language-independent as possible and has shown itself to be sufficient for Croatian texts in general, as evaluation results will show in Chapter 6. However, as was mentioned previously, HeidelTime's original rule set fails to normalise decades properly. To mend this, a simple post-processing module was developed, since multiple such expressions appeared in the tagged corpus.

Algorithm 3 Post-processing Decades

```
for timex3 in CAS do
  if timex is decadeExpr then
    value ← timex3
    decade ← functionArg(value)
    normalisedValue ← "century(value)decade" /*Concatenation*/
    setValue(timex3, normalisedValue)
  end if
end for
```

Instead of normalising the token "40-e" – the '40s – to "194", as HeidelTime would do for other languages, the Croatian rule would normalise it to "ÙNDEF-year-00-00 decadeCalc(4)". The post-processing module would normalise it into its proper decade using a simple algorithm outlined in Algorithm 3. The mechanism mentioned is dependent on language only for year and decade expression recognition and normalisation, and as such could be easily implemented to work for other languages HeidelTime supports. This extension, although theoretically extensible to any decade, currently works only for years and decades in the range 1000 – 2999, due to the way century and year recognition are defined in the resource files.

5. WikiWarsHR corpus

Most publicly available corpora are based on news articles. Such a source, while containing a number of temporal expressions, has some flaws. News articles are usually relatively short, so the number of expressions *per article* is rather low. This results in a limited temporal discourse structure and results in very few under-specified and relative temporal expressions, which are the primary obstacle of today's temporal expression extraction. Furthermore, there is no corpus of this type for Croatian that could serve as a standard for annotator evaluation. The WikiWarsHR corpus attempts to handle these issues for the Croatian language.

The WikiWarsHR corpus is a corpus of tagged historical texts from the Croatian Wikipedia, conceptually based on (Mazur and Dale, 2010). These articles were originally chosen (by Mazur an Dale) for their length, richness of temporal expressions and to avoid copyright issues. WikiWarsHR decided to follow their example. Twenty-two articles were taken from the Croatian Wikipedia (or, in one case, the Serbo-Croatian Wikipedia) for the corpus. These texts were pre-processed to leave only the plain text of the article, fitted into a standard SGML format and tagged manually.

5.1. Selecting the data

The selected articles mostly correspond to the original articles chosen for WikiWars. They were selected by querying Google with the phrases "most famous wars in history" and "biggest wars", then choosing the top results, eliminating duplicates and inappropriate articles, or those that could not be found, getting a set of 22 articles. WikiWarsHR contains the same articles, with two exceptions:

- The article on the Punic Wars was too short, so it was combined with the three separate articles about the three Punic Wars;
- The article on the Nigerian Civil War did not exist on the Croatian Wikipedia and was replaced with the Afghanistan War.

The Afghanistan War was added to the list according to this metric: A war had to be amongst the top results of a Google query for the Croatian phrase 'Moderni ratovi" (Modern Wars) and had to be long enough. The first criterion is due to the fact that most of the corpus refers to conflicts in the early 20th century or before, and as such contained very little time-based temporal expressions. As such, a more modern article, with references to specific hours and times of the day, was desirable to complete the corpus. The second is simply due to the fact longer articles are richer with temporal expressions. Two wars made the top of this list by these criteria: the Croatian Homeland War and the Afghanistan War. The first was longer, but due to its ideological tone, it was neglected in favour of the second one.

All articles but one were taken from the Croatian Wikipedia. The exception is the Polish-Soviet War, which did not exist on the Croatian Wikipedia. Instead, an article from the Serbo-Croatian Wikipedia was chosen. Due to the similarity between the languages, the only change which needed to be performed was replacing the Serbo-Croatian month names with their Croatian variants.

5.2. Pre-processing

The pre-processing of these articles was broken into two steps:

First, all links, image captions, the title, table of content and references were removed, leaving only the plain text of the article. This step removed a small number of temporal expressions present in image captions and the table of contents. However, these captions were often out of order with the flow of the text and could have undesirable effects on the focus of the article. The text was then wrapped into an SGML document containing these tags:

- DOC — The root tag of a well-formed document
- DOCID —- The document identification, in the form 06_KoreanWar
- DOCTYPE — A document type description. Constant as: <DOCTYPE source=wikipedia"> HISTORICAL ARTICLE </DOCTYPE>
- DATETIME — The date and time of document creation, in the ISO 8601 format, for a document creation time reference
- TEXT – The pre-processed text of the article

These documents form the input set and are available with the documentation, without any tags.

5.3. Tagging

The annotation language chosen was TIMEX3, a subset of the TimeML markup language. This differs from the original WikiWars corpus and its offspring corpora language of choice, TIMEX2. The primary reason for this was that HeidelTime outputs TIMEX3 and part of this corpus' purpose was to serve as a body of text for the development and testing of HeidelTime's implementation for Croatian. TIMEX3 is more verbose than TIMEX2, but only a limited subset of it was used. Events were not tagged (e.g. 'The start of World War II"), nor were references to focus used.

The corpus was hand-annotated, using the Callisto annotator tool and a custom-written minimal TIMEX3 DTD (*Document Type Definition*). The corpus was later annotated with HeidelTime and the missed annotations were added to the corpus, to cover expressions overlooked by the annotator ("annotator blindness"). Doing the reverse – first using an automated annotator and then adjusting by hand – would have been quicker and simpler, but would defeat the purpose of using this corpus as a development and testing aid for HeidelTime's Croatian implementation. The annotation was performed by two annotators. Annotator 1 annotated half of the corpus documents, while Annotator 2 annotated the second half. The two then annotators revised each others' work, adding missed annotations and discussing and correcting errors. There was no overlap between the sets annotated by the two annotators, as it was not deemed necessary, due to the simplicity of the vast majority of expressions. This was much quicker, but might have resulted in lower-quality annotations. The resulting corpus was stored and is available in two formats:

- Callisto's own AIF XML format;
- Original text enriched with inline annotations.

Any of the two formats can be used, if one is using the Callisto annotation tool and has access to the DTD definition used in the corpus, to reproduce the other one.

5.4. Corpus Statistics

The corpus consists of contains almost 60,000 tokens and 1,440 temporal expression, annotated in TIMEX3 format. This is about half the amount of tokens and timexes of the original WikiWars corpus. Individual statistics are presented in Table 5.1. The documents vary considerably in size, the smallest containing merely 235 tokens, considerably smaller than anything in the original WikiWars corpus, whereas the largest

contains 9722 tokens. That is also true for the density, the number of timexes per token, which range between a mere 0.016 (1.6 timexes per a hundred tokens) up to 0.055 (5.5 timexes per hundred tokens). The average density is 0.029, but most of it comes from a handful of documents.

An annotated excerpt from the article on the Soviet-Afghan war is given as an example below:

<TIMEX3 tid="t28" type="DATE" value="1979-12-27">*27. prosinca 1979.****</TIMEX3>****, 700 vojnika, uključujući 54 pripadnika specijalnih snaga KGB iz odreda Alfa i Zenit, preobučenih u afganistanske uniforme, zauzima važne vladine, vojne i medijske zgrade u Kabulu, uključujući i primarnu metu – Tadžbegovu palaču, gdje su uklonili predsjednika Amina. Operacija je počela u* ***<TIMEX3 tid="t29" type="TIME" value="1979-12-27T19:00:00">****19:00****</TIMEX3>****, kada su sovjetske specijalne snage (iz grupe Zenit) raznijele kabulski komunikacijski čvor, paralizirajući time afganistansko vojno zapovjedništvo. U* ***<TIMEX3 tid="t30" type="TIME" value="1979-12-27T19:15:00">****19:15****</TIMEX3>*** *je počeo juriš na Tadžbegovu palaču koji je trajao* ***<TIMEX3 tid="t31" type="DATE" value="PT45M">****45 minuta****</TIMEX3>****.* ***<TIMEX3 tid="t32" type="DATE" value="PRESENT_REF">****Istovremeno****</TIMEX3>*** *su zauzeti i ostali objekti (na primjer Ministarstvo unutarnjih poslova u* ***<TIMEX3 tid="t33" type="TIME" value="1979-12-27T19:15">****19:15****</TIMEX3>****).*
Operacija se završila sa potpunim uspjehom ***<TIMEX3 tid="t34" type="TIME" value="1979-12-28TMO">****ujutro 28. prosinca****</TIMEX3>****. Sovjetsko vojno zapovjedništvo u Termezu je objavilo na radio Kabulu da je Afganistan oslobođen Aminove vladavine.*

5.5. Remarks

As can be seen, it is a handful of articles which make the main volume of the corpus, both in length and in number of temporal expressions. Many articles are very brief, and some of them only outline the conflicts, instead of going through the conflicts and events in the war, resulting in a low number of temporal expressions. Quantity could be gained by steering further away from the original corpus and picking different articles. However, the Croatian Wikipedia has only a handful of articles of the type which can even be compared to their counterparts in English, German, or other major languages.

Wikipedia articles are written by a large number of people with various degrees of fluency and experience in writing and editing. As such, there were irregularities

in the text, such as typographical errors, for example the month July being written as “Spranj” instead of “Srpanj’. These errors were not corrected, but the correct semantics were inferred and marked by the annotators.

Table 5.1: Corpus Statistics

Document ID	Tokens	TIMEX3	$\frac{Tokens}{TIMEX3}$
01_WW2	9,239	194	47.62
02_WW1	5,958	187	31.86
03_AmCivWar	9,722	181	53.71
04_AmRevWar	3,068	76	40.37
05_VietnamWar	1,228	27	45.48
06_KoreanWar	2,369	100	23.69
07_IraqWar	5,524	162	34.10
08_FrenchRev	4.244	91	46.64
09_GrecoPersian	7,725	126	61.31
10_PunicWars	964	28	34.43
11_ChineseCivWar	272	12	22.67
12_IranIraq	1,065	22	48.41
13_RussianCivWar	822	15	54.80
14_FirstIndochinaWar	311	15	20.73
15_MexicanRev	235	13	18.08
16_SpanishCivilWar	1,049	22	47.68
17_AlgerianWar	573	20	28.65
18_SovietsInAfghanistan	2052	56	36.64
19_RussoJap	450	19	23.68
20_PolishSoviet	292	8	36.50
21_AfghanWar	1,498	34	44.06
22_SecondItaloAbyssinianWar	1,256	32	39.25
Total for Corpus	59,916	1,440	41.61
Average per Document	2,724	65	–
Standard Deviation	2,936	62	–

6. Experimental evaluation

6.1. Corpus used

The WikiWarsHR corpus was split into three parts for the purposes of rule development and evaluation. Two of them were used, while the third was discarded. They are:

- Articles 9_GrecoPersian and 10_PunicWars were discarded;
- Odd-numbered articles were used for development purposes;
- Even-numbered articles were used for testing purposes.

The articles describing the Greco-Persian and the Punic wars were not used because they contain dates from the previous era. While a system could have been developed to process these dates, a system for that was already being developed at the University of Heidelberg, the source of the original HeidelTime engine and resources, at the time. It was decided a system developed in parallel would be redundant.[1] However, this reduces the number of tokens by 8,689 and the number of temporal expressions by 154, significantly shrinking the corpus.

That left 10 articles for the development and the testing set each. Table 6.1 shows how they were divided. The articles are about equally split according to period of time, but the development set is slightly larger. The development set contains 29,563 tokens and 677 temporal expressions, while the testing set only has 21,644 tokens and 609 temporal expressions. Many articles were similar in style and form of expressing time and date, which positively influenced the evaluation results.

6.2. Evaluation

For the purposes of evaluation, the untagged input set files were combined into two large files – one for the development set, one for the test set. The same was done with

[1] It has since been developed and made public on the 27th of May 2014, at the website https://code.google.com/p/heideltime/

Table 6.1: Development and Testing Sets

Development	Testing
01_WW2	02_WW1
03_AmRevWar	05_AmCivWar
05_VietnamWar	06_KoreanWar
07_IraqWar	08_FrenchRev
11_ChineseCivWar	12_IranIraq
13_RussianCivWar	14_FirstIndochinaWar
15_MexicanRev	16_SpanishCivWar
17_AlgerianWar	18_SovietsInAfghanistan
19_RussoJap	20_PolishSoviet
21_AfghanWar	22_SecondItaloAbyssinianWar
29,563 tokens	21,644 tokens
677 timex	609 timex

the matching documents with inline tags. Each combined input document was then processed by HeidelTime. The output given by HeidelTime was then compared to the manually annotated text. The metrics used are *precision*, *recall* and *F1*. Precision measures how many of the matches were correct, while recall measures how many matches to gold annotations were found. For example, a text contains 30 temporal expressions. A tagger identifies 24 temporal expressions, but only 20 of these are valid matches, while the remaining four are false positives. The precision would then be $20/24$, while the recall would be $20/30$. The F1-score is a measure of accuracy computed from precision and recall defined as:

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

Three levels of evaluation were considered, all using these three scores. First, only the identification phase was considered, without considering the normalised value. In this step, we try to see if a temporal expression was recognised at all. In the next level, we also consider the expression's type (out of Date, Time, Duration and Set) and its correctness. Finally, the third level of evaluation compares the normalised value, as

well. Each level is evaluated for correctness only if the level before has evaluated to correct. As such, if a matches' type was incorrect, its value will not be considered at all.

These scores were computed in two settings: relaxed and strict. These modes of operation refer to the way identification is evaluated. In relaxed mode, a match can be only partial, but will still evaluate as a correct match. For example, if we have an expression "in late June 1994" and only the word "June" was recognised to be part of a temporal expression, it would still pass evaluation. In strict mode, however, a match has to be exact – the annotations have to begin and end at the same points. While this does not directly apply to the type and value scores, it still affects their scores. If only a partial match was found, its value might still be correct. In relaxed mode, the type and value would be considered for the previous example, but in strict mode they would be automatically discarded as incorrect, even if a correct value was inferred by some other means.

Table 6.2: Development Set Evaluation Results

Relaxed			
Score	Identification	Type	Value
Precision	0.9508	0.9392	0.8611
Recall	0.9705	0.9586	0.8789
F1	0.9605	0.9488	0.8699
Strict			
Score	Identification	Type	Value
Precision	0.9320	0.9291	0.8582
Recall	0.9513	0.9483	0.8759
F1	0.9415	0.9386	0.8670

Given those evaluation methods, evaluation was performed on the development and the test set. Results are visible in Table 6.2 and Table 6.3, respectively. The results are very good, especially for temporal expression identification and type normalisation. They are comparable, and in some cases better, than the results HeidelTime achieves for other languages. Furthermore, the relaxed and strict results are almost negligible. This shows that the crafted rule set almost always made a correct match. Furthermore, the results for the development and testing set are very close.

Although some of this is due to the number and quality of rules implemented for Croatian, there are a few important factors to consider. A great majority of expres-

sions in these texts are very simple – complete years or dates, or seasons and months referring to a recently mentioned year. Such cases are easily handled by HeidelTime's focus resolving methods. Furthermore, some articles contained time-line summaries, which were simple to identify and normalise and contained many temporal expressions, thus boosting the score. Additionally, many articles were very similar, likely at least partially written by the same group of authors. This resulted in similar expressions across articles and enabled a small subset of rules to evaluate many temporal expressions correctly. Finally, much of the annotation was performed by the same person who developed the rule set. Knowledge of the rules could result in unintentional creation of similar annotations and vice-versa. An evaluation performed on a similar, but independently annotated corpus would likely have slightly lower results. However, even with these facts taken into consideration, the results are remarkably high for all three measured parameters.

Table 6.3: Test Set Evaluation Results

Relaxed			
Score	Identification	Type	Value
Precision	0.9436	0.9308	0.8583
Recall	0.9622	0.9491	0.8752
F1	0.9528	0.9398	0.8667
Strict			
Score	Identification	Type	Value
Precision	0.9163	0.9147	0.8486
Recall	0.9343	0.9327	0.8654
F1	0.9252	0.9236	0.8569

6.3. Error Analysis

As can be seen in the evaluation results, most errors occur in value normalisation. To help determine and analyse the cause for these errors, the evaluation engine was written to have a verbose mode of operations – each time a mismatch occurred, it would determine the cause and print out a message detailing the fault. This has shown the majority of errors stem from incorrect focus. For example, see the following paragraph from the Croatian Wikipedia article on the Iraq War, with temporal expressions marked in bold:

> ***31. siječnja 2005.*** *održati su prvi poslijeratni izbori kako bi se izabrali predstavnici za prijelazno novo iračko vijeće od 275 članova koje bi trebalo donijeti stalni ustav. Usprkos tome što je na biračkim mjestima bilo nasilja, a Suniti su bojkotirali izbore, većina Kurda i Šijita je sudjelovalo i izbori su bili razmjerno uspješni. Oko 8,4 milijuna ljudi je glasovalo u, kako je ocijenjeno, "prvim demokratskim izborima Iraka"* ***4. veljače*** *Paul Wolfowitz je objavio da će se oko 15.000 američkih vojnika, koji su osiguravali red na izborima, vratiti natrag kući.* ***Veljača****,* ***ožujak*** *i* ***travanj*** *su bili relativno mirni mjeseci u usporedbi s krvavim* ***studenim*** *i* ***siječnjom****,*
> ...

The text refers to January, February and March of 2005. However, the underlined expression, *studeni* refers to November 2004, but the focus is still on year 2005, causing a wrong value normalisation result. Such situations often occur in this corpus. Likewise, the narrative sometimes switches from the end of the year to the beginning of the new year – one sentence refers, for example, to December 1940, while the next refers to January 1941, but without mentioning the year explicitly. It is implicitly clear to a human that the focus has shifted, but HeidelTime's current focus-determining system cannot resolve this issue. Another error of this type is common when the subject is switched, implicitly switching the temporal focus. For example, in the article about World War II, a paragraph details the events on one front throughout multiple years, then switches to another front, starting the narrative from the beginning of the war again. While the text might mention it (for example stating "at the same time as the beginning of the xx offensive"), there is no explicit expression which HeidelTime can use to switch focus. To make matters worse, these kinds of errors usually propagate through multiple sentences or even paragraphs.

A secondary source of errors, primarily in the identification phase, are unique expressions. Some complex expressions appear only once and are unlikely to be common, so no rules were implemented for them. However, a large enough number of such unique expressions lowers the score somewhat. One example of such an error can be seen in the the Iraq War article: "nedjelju oko 14,45 sati po srednjoeuropskome vremenu". Most of the errors in the identification phase were caused by such errors. While this could be mended, it was decided by the author that the rule set was already large enough and should not be expanded to cover these cases.

This shows a major weakness of rule-based approaches in general – if a rule or some sort of inferrence mechanism doesn't exist for a specific case, it will not be an-

notated and normalised. Expanding a rule-based system to work with different corpora and in different domains of text therefore results in a growing number of rules or extreme abstraction of rules.

7. Conclusion

This thesis presents a usable resource set for using HeidelTime for texts in Croatian, as well as WikiWarsHR, a corpus of historical narratives annotated with the TIMEX3 annotation markup language. The results of the first can be used in higher-level natural language analysis applications, while the second might prove useful in the development and evaluation of other temporal expression taggers.

HeidelTime's engine, combined with the hand-crafted set of rules, handles well, achieving exceedingly excellent results on the WikiWarsHR corpus, both for identification and normalisation. Its internal mechanisms handle major obstacles in the field, such as focus and underspecification, in all but the most difficult or ambiguous cases. The results are comparable to, if not better than, HeidelTime implementations for other languages on similar corpora. However, there is still space for improvement. The rule set could be reduced to a more compact form, or expanded for better results in this or other corpora. Furthermore, a historical date module was implemented for HeidelTime, but too late to be included in this thesis. Such an extension would allow the Croatian implementation of HeidelTime rules to conform to the standard.

Additionally, HeidelTime could be adjusted to work with lemmas. This would reduce the number of expression and normalisation resources necessary for Croatian, as well as other heavily inflected languages. As this would require extensive changes on the HeidelTime engine itself, it was not performed in the scope of this thesis.

The WikiWarsHR corpus offers a sizeable number of annotated temporal expressions. Although it is much smaller than corpora in other languages, such as TimeBank or WikiWars, it contains nearly 60,000 tokens and almost 1,500 tagged temporal expressions. It is also fundamentally different in type – historical narratives, rather than news articles – than most available corpora, especially for Croatian. Its size could be increased if a choice was made to search for articles that were longer and richer with temporal expressions. However, a choice to conform more closely to the original WikiWars corpus was made instead.

The use of TIMEX3 as an annotating standard in both corpus construction and

automatic timex tagging might prove to be useful – the TIMEX3 annotations could be supplemented with other TimeML annotations, which is especially suitable for event extraction.

Although this thesis has brought no innovation to the fields of computer science and natural language processing, it provides the scientific community with a useful tool, usable in other, more complex, natural language processing procedures, such as document summarisation and question answering, as well as with a body of annotated texts in a relatively unexplored domain.

BIBLIOGRAPHY

Željko Agić, Nikola Ljubešić, and Danijela Merkler. Lemmatization and morphosyntactic tagging of croatian and serbian. U *Proceedings of ACL*, 2013.

David Ahn, Sisay Fissaha Adafre, and Maarten De. Recognizing and interpreting temporal expressions in open domain texts. 2005.

Branimir Boguraev, James Pustejovsky, Rie Ando, and Marc Verhagen. Timebank evolution as a community resource for timeml parsing. *Language Resources and Evaluation*, 41(1):91–115, 2007.

Angel X Chang and Christopher Manning. Sutime: A library for recognizing and normalizing time expressions. U *LREC*, stranice 3735–3740, 2012.

L. Ferro, L. Gerber, I. Mani, B. Sundheim, and G. Wilson. TIDES 2005 standard for the annotation of temporal expressions. Technical report, MITRE, September 2005a.

Lisa Ferro, Laurie Gerber, Janet Hitzeman, Elizabeth Lima, and Beth Sundheim. Ace time normalization (tern) 2004 english training data v 1.0. *Philadelphia, USA*, 2005b.

David Ferrucci and Adam Lally. Uima: an architectural approach to unstructured information processing in the corporate research environment. *Natural Language Engineering*, 10(3-4):327–348, 2004.

Jannik Strötgen Julian Zell Michael Gertz. Heideltime: Tuning english and developing spanish resources for tempeval-3. *Atlanta, Georgia, USA*, stranica 15, 2013.

Péter Halácsy, András Kornai, and Csaba Oravecz. Hunpos: an open source trigram tagger. U *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, stranice 209–212. Association for Computational Linguistics, 2007.

Bart Jongejan and Dorte Haltrup. the cst lemmatiser. *Center for Sprogteknologi, University of Copenhagen version*, 2, 2005.

Oleksandr Kolomiyets and Marie-Francine Moens. Meeting tempeval-2: shallow approach for temporal tagger. U *Proceedings of the Workshop on Semantic Evaluations: Recent Achievements and Future Directions*, stranice 52–57. Association for Computational Linguistics, 2009.

Pawet Mazur and Robert Dale. Wikiwars: A new corpus for research on temporal expressions. U *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, stranice 913–922. Association for Computational Linguistics, 2010.

Matteo Negri and Luca Marseglia. Recognition and normalization of time expressions: Itc-irst at tern 2004. U *TERN 2004 Evaluation Workshop*, 2004.

James Pustejovsky, José M Castano, Robert Ingria, Roser Sauri, Robert J Gaizauskas, Andrea Setzer, Graham Katz, and Dragomir R Radev. Timeml: Robust specification of event and temporal expressions in text. *New directions in question answering*, 3: 28–34, 2003a.

James Pustejovsky, Patrick Hanks, Roser Sauri, Andrew See, Robert Gaizauskas, Andrea Setzer, Dragomir Radev, Beth Sundheim, David Day, Lisa Ferro, et al. The timebank corpus. U *Corpus linguistics*, svezak 2003, stranica 40, 2003b.

Jannik Strötgen and Michael Gertz. Heideltime: High quality rule-based extraction and normalization of temporal expressions. U *Proceedings of the 5th International Workshop on Semantic Evaluation*, stranice 321–324. Association for Computational Linguistics, 2010.

Jannik Strötgen and Michael Gertz. Wikiwarsde: A german corpus of narratives annotated with temporal expressions. U *Proceedings of the conference of the German society for computational linguistics and language technology (GSCL 2011)*, stranice 129–134, 2011.

Jannik Strötgen, Ayser Armiti, Tran Van Canh, Julian Zell, and Michael Gertz. Time for more languages: Temporal tagging of arabic, italian, spanish, and vietnamese. *ACM Transactions on Asian Language Information Processing (TALIP)*, 13(1):1, 2014.

Naushad UzZaman and James F Allen. Event and temporal expression extraction from raw text: first step towards a temporally aware system. *International Journal of Semantic Computing*, 4(04):487–508, 2010.

Abstract

Temporal expression extraction has recently become a popular field of natural language processing. This task consists of locating temporal expressions and normalising them into a canonical form. Recent successes achieved by rule-based temporal taggers, HeidelTime in particular, and a lack of a good temporal tagger for Croatian have inspired the construction of HeidelTime rule set for the Croatian language. Additionally, WikiWarsHR, a corpus of Wikipedia-based historical narratives tagged with TIMEX3, has been developed, both for the purpose of developing the HeidelTime rule set and for future use. Results achieved by the Croatian implementation of HeidelTime are comparable to implementations for other languages, while WikiWarsHR is close in size and density of tags to other publicly available corpora.

Keywords: TimeML,TIMEX3, rule-based tagging, temporal expression normalisation, natural language processing.

Označavanje vremenskih izraza u tekstovima na hrvatskome jeziku

Sažetak

Označavanje vremenskih izraza u zadnje je vrijeme postalo jedno od fokusa obrade prirodnog jezika. Ono se sastoji od identifikacije vremenskih izraza i njihove normalizacije u kanonski oblik. Nedavni uspjesi označivaća baziranih na pravilima, a posebno sustava HeidelTime, kao i nedostatak uspješnih označivaća za hrvatski jezik, potaknuli su oblikovanje skupa pravila za hrvatski za HeidelTime. Stvoren je i korpus sastavljen od članaka s hrvatske Wikipedije baziranih na člancima o ratovima. Korpus je anotiran TIMEX3 oznakama, a njegova je svrha bila pomoć u razvoju skupa pravila, ali otvara i mogućnost korištenja za druge alate i svrhe. Implementacija pravila za hrvatski postigla je rezultate uspredive s rezultatima implementacija za druge jezike, a korpus WikiWarsHR usporediv je sa sličnim korpusima na drugim jezicima po duljini i broju označenih vremenskih izraza.

Ključne riječi: TimeML,TIMEX3, označavanje bazirano na pravilima, normalizacija vremenskih izraza, obrada prirodnog jezika.