|  |
| --- |
| TERNIP: Temporal Expression Recognition and Normalisation in Python |
| INTERIM REPORT  COM6920 Thesis Preparation |
| Christopher Northwood  *Supervisor: Mark Hepple* |

Table Of Contents

[Table Of Contents 3](#_Toc260675349)

[1 Introduction 4](#_Toc260675350)

[2 Background 5](#_Toc260675351)

[2.1 Temporal Expressions 5](#_Toc260675352)

[2.2 Annotation Standards 6](#_Toc260675353)

[2.3 Temporal Expression Taggers 7](#_Toc260675354)

[2.4 Evaluating Tagger Performance 9](#_Toc260675355)

[2.5 Rule Discovery 9](#_Toc260675356)

[3 Project Aims 10](#_Toc260675357)

[4 Work Plan 11](#_Toc260675358)

[4.1 Overview 11](#_Toc260675359)

[4.2 Python Tagger 11](#_Toc260675360)

[4.3 Automatic Rule Discovery 11](#_Toc260675361)

[4.4 Evaluation 11](#_Toc260675362)

[4.5 Timescale 12](#_Toc260675363)

[5 Bibliography 13](#_Toc260675364)

# Introduction

In this report, the outline of a system for temporal expression recognition and normalisation is presented, called TERNIP – Temporal Expression Recognition and Normalisation in Python.

Temporal expressions are words and phrases which refer to some point in time (Ahn, Rantwijk, & de Rijke, 2007), and the distinct, but related, tasks of recognition and normalisation refer to the identification and resolution of these expressions to some standard format in time. A more detailed overview of temporal expressions and the development of the fields of recognition and normalisation is given in section 2.1.

A number of systems and approaches have been taken to the tasks of recognition and normalisation, a survey of which appears in section 2.3. Additionally, a number of standards for annotation have been defined which is covered in section 2.2, and the Time Expression Recognition & Normalization evaluation (MITRE, 2004) provides a corpus and some comparative results, which is outlined in section 2.4.

TERNIP aims to build on work already done, specifically that of the GUTime tagger (Verhagen, et al., 2005), by building a system for the recognition and normalisation of temporal expressions in Python (van Rossum, 1995). The more detailed aims of the project are outlined in section 3.

Finally, section 4 proposes a series of tasks which will be undertaken by the project to meet the stated aims, along with a plan for the execution of these tasks.

# Background

## Temporal Expressions

Temporal expressions, or “timexes”, are “phrases or words that refer to times, where times may be points or durations, or sets of points or durations” (Ahn, Rantwijk, & de Rijke, 2007) , and the identification and interpretation of these timexes is an active topic of research.

Most systems deal with two distinct, but related, tasks for the identification of timexes. The first is that of recognition, which simply identifies which phrases in some text are temporal, that is, refer to some point in time. The second task is that of normalisation, which takes the identified expressions, and attempts to resolve it into some standard format (e.g., ISO 8601) to anchor the expression at a particular point in time (Ahn, Adafre, & de Rijke, Recognizing and Interpreting Temporal Expressions in Open Domain Texts, 2005).

Interest in recognition of temporal expressions grew out of the field of information extraction. The Message Understanding Conferences of the 1990s dealt with the tasks of named entity recognition, and early timex recognition systems simply dealt timex recognition as a part of named entity recognition (Krupka & Hausman, 1998). Temporal expression recognition is clearly an important task for information extraction; however identification of temporal expressions by itself is of limited usefulness.

Normalisation is important to allow for further processing, such as construction of event chronologies, or in question answering systems, and is an important part of natural language understanding. In the phrase “Do you want to go to the pub at 7?”, a human may normalise the expression “7” to a particular point in time based on context of the current date, and the background knowledge that visits to public houses are more likely in the evening.

(Mani & Wilson, 2000) was a prominent, early system which used a rule-based system for normalisation based on establishing tense. Following this, the Time Expression Recognition and Normalization (TERN) evaluation as part of the 2004 Automated Content Extraction (ACE) programme (MITRE, 2004) was the first competition that dealt specifically with recognition and normalisation as a distinct task from named entity recognition.

Following this early work and the TERN competition, interest in temporal expressions has grown, with multiple systems and approaches to recognition and normalisation taken. These systems and approaches are discussed further in section 2.3.

Simple normalisation of temporal expressions is not enough to capture the full range of temporal information available in a body of text, as much temporal information is implicit (Verhagen, 2004). For example, in the phrase “A goal was scored shortly after kick-off”, there is no explicit temporal information there, but there is some implicit information that could be obtained. In this case, the events of the goal being scored and kick-off are identified, and there is a temporal ordering between them, as well as implicit temporal information in these events themselves.

Much recent research has been done on identifying and annotating temporal relations, which build on top of temporal expression recognition and normalisation, however effective temporal recognition and normalisation is still required for this work to be effective.

## Annotation Standards

A number of standards for annotation of temporal expressions have emerged over time. The first annotation formats were typically based on SGML and XML and were simply in a format decided by the tagger. Over time, a standardisation effort for annotation emerged, culminating in TimeML (Pustejovsky, et al., 2003). TimeML is an XML-based annotation language, complete with a set of guidelines for timex annotation, based on the earlier TIDES standard (Ferro, Mani, Sundheim, & Wilson, 2001) and work by (Setzer, 2001).

Of most interest to this project in the TimeML specification is the TIMEX3 tag, which extends the annotation functions of the earlier TIMEX (Setzer, 2001) and TIMEX2 (Ferro, Mani, Sundheim, & Wilson, 2001) tags.

The TIMEX3 tag is used to represent time expressions, and a number of attributes are used to define this. The most important attribute is the ‘value’ attribute, based on the TIMEX2 ‘val’ attribute, which is used to hold the either the normalised time, or an unanchored duration. This value can either be a simple string referencing a specific time, a pair of strings separated by a slash representing a duration anchored in specific points of times, or a simple string representing an unanchored duration.

The format used for denoting dates is based on the modifications of ISO 8601 described in the TIDES standard (Ferro, Mani, Sundheim, & Wilson, 2001), with a number of modifications. As natural language temporal expressions allow a differing degree of precision, the TimeML standard allows for unknown components of a date to be replaced with the character ‘X’ (e.g., XXXX-05-03 represents May 3rd, when the year is unknown). Expression values are also omitted from right-to-left to the appropriate level of precision (e.g., 2010-05 for May 2010, but 2010-05-XX for ‘a sunny day in May 2010’).

To support further imprecision in natural language expressions that the ISO 8601 standard does not specify, TIDES, and subsequently TimeML, specify a number of replacement components which can be used as values in particular components of an ISO 8601 expression. This includes tokens such as “DT” in the hour position to represent “day time”, “WI” in the place of month to represent “winter” and “WE” in the place of a day to represent “weekend”.

In addition to these modified ISO 8601 values, a number of tokens are also allowed in the value attribute when expressions can not be resolved to a timestamp, for example “PRESENT\_REF” for time expressions such as “currently”, “FUTURE\_REF” for “future” and “PAST\_REF” for “long ago”.

The second TIMEX2 attribute adopted by TIMEX3 is the MOD attribute, which is used for timexes that have been modified in natural language in such a way that can not be expressed by value alone. These modifiers modify points in time and durations, allowing for expressions such as “before June 6th”, “less than 2 hours long”, or “about three years ago” to be correctly expressed.

TimeML’s TIMEX3 tag does not directly incorporate the other attributes of TIMEX2, but captures the information in other ways. One such attribute is the “functionInDocument” optional attribute which indicates whether or not this tag is providing a temporal anchor for other timexes in the document. The values this attribute can take come from the PRISM standard (IDEAlliance, 2008) and denote that a timex can take functions such as creation time, publication time, etc. The PRISM standard is typically used to mark up metadata to a document, rather than directly dealing with the content itself, whereas TimeML expands this to allow the content of the document to be tagged with these functions.

TimeML also allows a timex to be annotated as a “temporal function” (e.g., “two weeks ago”), and supplies a number of attributes to support the capturing of this data. Similarly, more attributes are provided to denote quantified times (such as “twice a month”), and to anchor durations to other timexes.

As interest in temporal expressions has grown to include event identification and temporal relations, the TimeML standard also includes tags and annotation guidelines for more than just timexes, such as events, signals for determining interpretation of temporal expressions and dependencies between these events and times.

In addition to the formal specification of TimeML, a set of annotation guidelines is published (Saurí, Littman, Knippen, Gaizauskas, Setzer, & Pustejovsky, 2006), which contains information on when an expression should be tagged, and how the attributes should be filled, in order to ensure consistency between TimeML annotated documents. For the TIMEX3 tag, these are mostly inherited from the TIMEX2 guidelines, which are built on top of two basic principles (Ferro, Mani, Sundheim, & Wilson, 2001):

1. “If a human can determine a value for the temporal expression, it should be tagged”
2. “VAL must be based on evidence internal to the document that is being annotated”

The TIMEX2 guidelines then continue to specify a number of situations where a timex should be tagged, including fairly detailed indicators to trigger a tag, and where it should not. One rule it gives relates to proper nouns, where any temporal expression incorporated within (e.g., the terrorist group “Black September”) should not be tagged, and a proper noun treated as an atomic unit. Additionally, specific rules are given to the extent of a tag, for example, when a temporal expression includes premodifiers (as handled by the ‘mod’ attribute), the premodifiers should be part of the tagged text.

The annotation guidelines for TIMEX2 also include guidelines for the format of the expected output tag (particularly for the form of the value attribute), depending on the type of expression that was recognised.

The TimeML rules extend these TIMEX2 guidelines, usually as a result of changes in the TIMEX3 tag from the TIMEX2 tag. These include changes in tagging extent recommendations for expressions embedded within each other, and for postmodifiers.

Additionally, TimeML also allows for empty TIMEX3 tags, which can be used to denote implicit timexes in text, often for anchored durations.

As with the wider TimeML standard, the annotation guidelines additionally define how to annotate events, signals and relations, however as this project focuses on the annotation of timexes only, they are not considered here.

## Temporal Expression Taggers

Temporal expression taggers are tools which annotate the timexes in some input text. Early taggers focussed just on identification with normalisation becoming incorporated into later systems.

<<< Investigations to be done on hand-tagging: Setzer and Rob G >>>

The tasks of automated recognition and normalisation is often rolled into the same tool, although (Ahn, Adafre, & de Rijke, 2005) argue that separation of these components is beneficial.

<<< typically marked up with POS tags? >>>

<<< Recognition trivial, grew out of entity recognition >>>

The earliest automated temporal expression annotation systems treated temporal expression recognition as a task along with entity recognition , and used simple hand-written rules . In both systems, grammars were provided for the named entity recognisers and the time expressions simply recognised. No normalisation was performed in these early systems.

In English, temporal expressions are recognised as being highly idiosyncratic, but attempts have been made by linguists to make generalisations of the underlying grammar (Flickinger, 1996). This underlying concept, that the idiosyncrasies can be captured by some generalised rules, is used by automated rule-based annotators.

(Pustejovsky, et al., 2003) identifies three main classifications of temporal expressions:

* “Fully-specified temporal expressions (e.g., June 11, 1989, or Summer 2002);
* Underspecified temporal expressions (e.g., Monday, next month, two years ago);
* Durations (e.g., three months, two years).”

The recognition task is generally considered to be “do-able” (Ahn, Adafre, & de Rijke, Recognizing and Interpreting Temporal Expressions in Open Domain Texts, 2005), with two main approaches to the task: rule-based and machine learning based. Unlike recognition, normalisation is considered a more difficult task, especially for underspecified temporal expressions, and durations.

<<< Normalisation has issues, due to ambiguity, granularity of time, incompleteness of data, etc... Discuss how ambiguity is often handled in different systems. >>>

Becoming frequently common is for the tagger to be incorporated into a larger toolkit that deals with temporal relations <<< such as TARSQI – likely to happen as (Ahn, Adafre, & de Rijke, Recognizing and Interpreting Temporal Expressions in Open Domain Texts, 2005) sensibly suggest separation of roles >>>

<<< Overview of current state of the art for taggers, then introduce individual taggers and the technologies behind them For each one outline system and annotation scheme used

Automated annotation based on hand-written rules

* Mani and Wilson 2000 (hand-written rules for expression tagging) – tempex (basis of GUtime)
* Tarqas (as a component)
* Tarsqi (uses GUTime as a component)
* DANTE
* Chronos
* Saquete
* Mikheev, et al; Krupka, Hausman(early work)

Automated annotation based on machine learning

Ahn, Rantwijck blah includes a particular architecture for machine learning which may be useful

* Mani and Wilson 2000 also includes a discussion of machine learning
* Kolomiyets...
* Hacioglu – SVM
* Baldwin
* Jang, Baldwin, Mani >>>

### GUTime

Talk about GUTime specifically

## Evaluating Tagger Performance

Contests for temporal expression recognition date back as far as the Message Understanding Conference of 1995, but only as part of a broader named entity recognition task. In 2004, the Automated Content Extraction (ACE) programme launched the Time Expression Recognition and Normalization (TERN) evaluation sub-task (MITRE, 2004), which focussed on two sorts of systems – those that do recognition only, and those that do recognition and normalisation.

Although both TIDES (Ferro, Mani, Sundheim, & Wilson, 2001) and TimeML (Pustejovsky, et al., 2003) define annotation guidelines for the TIMEX2 and TIMEX3 tags respectively, the competitions also define additional guidelines which were used for the hand-tagging of the datasets.

<<< Talk about TERN (need TERN dataset), TimeBank (have 1.1 – need 1.2), TempEval, AQUAINT. TempEval is for relations, TempEval-2 more interesting >>>

## Rule Discovery

<<< More research needed. Want to look into automated rule discovery Adaptive Information Extraction from Text by Rule Induction and Generalisation >>>

# Project Aims

# Work Plan

## Overview

High level work plan (introduction to different components, etc) about half a page

## Python Tagger

### Overview

Talk about how we’re going to port GUTime, etc

### Workload

|  |  |
| --- | --- |
| Task Name | Estimated Workload |
|  |  |
|  |  |
|  |  |
|  |  |

## Automatic Rule Discovery

### Overview

Talk again (about 2/3rds of a page) about how this is going to work

### Workload

|  |  |
| --- | --- |
| Task Name | Estimated Workload |
|  |  |
|  |  |

## Evaluation

### Overview

Talk about how the evaluation of my system is going to work (basically, as a TERN project)

### Workload

|  |  |
| --- | --- |
| Task Name | Estimated Workload |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

## Timescale

# Bibliography

Ahn, D., Adafre, S. F., & de Rijke, M. (2005). Recognizing and Interpreting Temporal Expressions in Open Domain Texts. In S. N. Artëmov, H. Barringer, A. S. d'Avila Garcez, L. C. Lamb, & J. Woods (Eds.), *We Will Show Them: Essays in Honour of Dov Gabbay* (Vol. 1, pp. 31-50). College Publications.

Ahn, D., Rantwijk, J. v., & de Rijke, M. (2007). A Cascaded Machine Learning Approach to Interpreting Temporal Expressions. *Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics* (pp. 420-427). Rochester, New York, USA: The Association for Computational Linguistics.

Flickinger, D. (1996). English time expressions in an HPSG grammar. *In Studies on the Universality of Constraint-Based Phrase Structure Grammars Gunji* , 1-8.

Krupka, G. R., & Hausman, K. (1998). IsoQuest Inc.: Description of the NetOwlTM Extractor System as Used for MUC-7. *In Proceedings of 7th Message Understanding Conference (MUC-7).*

Mani, I., & Wilson, G. (2000). Robust temporal processing of news. *ACL '00: Proceedings of the 38th Annual Meeting on Association for Computational Linguistics* (pp. 69-76). Morristown, NJ, USA: Association for Computational Linguistics.

Mikheev, A., Grover, C., & Moens, M. (1998). Description of the LTG system used for MUC-7. *In Proceedings of 7th Message Understanding Conference (MUC-7).*

MITRE. (2004). *Time Expression Recognition and Normalization Evaluation*. Retrieved April 28, 2010, from http://fofoca.mitre.org/tern.html

Pustejovsky, J., Castaño, J., Ingria, R., Saurí, R., Gaizauskas, R., Setzer, A., et al. (2003). TimeML: Robust Specification of Event and Temporal Expressions in Text. *IWCS-5, Fifth International Workshop on Computational Semantics* .

van Rossum, G. (1995). *Python Reference Manual.* CWI Report CS-R9525.

Verhagen, M. (2004). *Times Between the Lines.* Waltham, MA, USA: Brandeis University.

Verhagen, M., Mani, I., Sauri, R., Knippen, R., Jang, S. B., Littman, J., et al. (2005). Automating temporal annotation with TARSQI. *Proceedings of the ACL 2005 on Interactive poster and demonstration sessions* (pp. 81-84). Ann Arbor, Michigan: Association for Computational Linguistics.