TERNIP: Temporal Expression Recognition and Normalisation in Python

INTERIM REPORT

COM6920 Thesis Preparation

Christopher Northwood

Supervisor: Mark Hepple

TABLE OF CONTENTS

1 INTRODUCTION

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

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.

Interest in temporal expressions arises from their obvious utility, both within the wider fields of linguistics and philosophy, and as a task within the natural language processing field. Recognition and normalisation of temporal expressions in natural language text is clearly an important task for humans to be able to function in modern society (for example, correctly recognising and normalising the temporal expression in the sentence "Shall we meet next Wednesday at 6pm?").

Temporal expressions in natural language are very rich and are often ambiguous (for example, in the phrase "midnight on Tuesday", as midnight refers to the instant between two days, it is unclear whether this refers to the time between Monday and Tuesday, or Tuesday and Wednesday).

In the field of natural language understanding, being able to handle temporal expressions in a similar way is clearly desirable. For example, in automatic summarisation of news texts, the ability to construct a chronology of events aids in this task.

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 have been developed and different approaches have been taken to the tasks of recognition and normalisation, a survey of which appears in section 2.5. Additionally, a number of standards for annotation have been defined, which are covered in section 2.2, and the Time Expression Recognition & Normalization (TERN) evaluation (MITRE, 2004) provides a corpus and some comparative results, which is outlined in section 2.3.

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.

2 BACKGROUND

2.1 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. Temporal expressions are a very rich form of natural language, with Pustejovsky, et al. (2003) identifying three main types 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)."

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 them into some standard format (e.g., ISO 8601) to anchor the expression at a particular point in time (Ahn, Adafre, & de Rijke, 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 task of named entity recognition, and early timex recognition systems simply treated 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 recognise the expression "7" as a temporal expression and normalise that 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) introduced a prominent system which used a rule-based system for recognition and normalisation using the technique of 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 built and many approaches to recognition and normalisation investigated. These systems and approaches are discussed further in section 2.5.

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 focussed on identifying and annotating temporal relations, a task which builds on top of temporal expression recognition and normalisation; however a high performing temporal recognition and normalisation system is still required for this work to be effective.

2.2 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 in 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. An example of this tag is shown in Sample 1.

INDEPENDENCE, Mo. _ The North Atlantic Treaty Organization embraced three of its former rivals, the Czech Republic, Hungary and Poland on <TIMEX3 tid="**t3**" type="**DATE**" functionInDocument="**NONE**" temporalFunction="**true**" value="**1999-03-12**">**Friday**</TIMEX3>, formally ending the Soviet domination of those nations that began after World War II and opening a new path for the military alliance.

SAMPLE 1 - A SAMPLE TIMEX3 TAG FROM THE AQUAINT CORPUS (VERHAGEN & MOSZKOWICZ, 2008)

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 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 handle, 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 cannot 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 cannot be expressed by value alone. These modifiers alter 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 has been published (Saurí, Littman, Knippen, Gaizauskas, Setzer, & Pustejovsky, 2006), which contains information about 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 of when to and when not to trigger a tag. 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.

2.3 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 perform recognition only, and those that perform both 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 gold standard datasets. Issues with inter-annotator agreement were identified by (Setzer & Gaizauskas, 2001), so the purpose of these additional guidelines is to ensure high inter-annotator agreement.

The TERN contest defines system performance by using f-measures against different metrics of the system. An f-measure, sometimes referred to as an F1 score, is the harmonic mean of precision and recall. Precision is a measure of relevance – that is, of all the identified timexes or normalised values, what proportion of those are true positives or accurate. Recall is a measure of retrieval – that is, of all the possible timexes or normalised values in the document, what proportion of these were identified.

The first metric TERN uses to measure performance is that of detection of temporal expressions. The second is to correctly recognise the extent of the temporal expression, and the third is to correctly normalise the temporal expression into some time. This final metric also can be subclassified into an absolute f-measure, which considers the performance of normalisation against all timexes, not just those recognised. Therefore, the absolute f-measure gives the headline metric for all parts of the system, whereas the breakdown allows performance of individual components to be considered. For the systems given below, we consider the recognition metric as both the recognition and extent detection tasks, and normalisation as the final, non-absolute metric.

Using these metrics, the best performing system for recognition is the ATEL system (Hacioglu, Chen, & Douglas, 2005), and for normalisation, Chronos (Negri & Marseglia, 2004). These systems, and others, are discussed further in section 2.5 below.

2.4 CORPORA

There are few publicly available corpora annotated with TIMEX tags. The TERN competition saw the creation of the TERN corpus, which consists of English and Chinese text annotated with TIMEX2 tags (Ferro, Mani, Sundheim, & Wilson, 2001). The texts that make up the TERN corpus are drawn from news articles. Performance on this corpus is typically used as a comparative measure between different systems.

The TimeBank corpus (Pustejovsky, et al., 2006) is a later corpus that extends the TERN corpus to use the TimeML standard (Pustejovsky, et al., 2003), and also includes additional documents, still from the news genre. Most recent contests use the TimeBank corpus as a base, although typically modify it for their specific needs (for example, in TempEval, a simplified form of TimeML was used).

A final corpus of note is the AQUAINT corpus (Verhagen & Moszkowicz, 2008), sometimes referred to as the 'Opinion' corpus. This also uses news texts and is annotated to the same specifications of the TimeBank corpus, although the annotation effort is not as mature. Efforts are underway to merge the AQUAINT and TimeBank corpora into a new, larger corpus with a higher annotation standard.

The corpora discussed above are not considered perfect. As Setzer & Gaizauskas (2001) showed, high inter-annotator agreement on temporal expressions is hard to come by. In the case of the TimeBank corpus, inter-annotator agreement for TIMEX3 tags is 0.83 for exact matches, or 0.96 for partial matches (average of precision and recall). Other tags are lower, but they are of limited interest for this project and, as such, are not considered.

2.5 TEMPORAL EXPRESSION TAGGERS

Temporal expression taggers are tools which annotate the timexes in some input text. The earliest automated temporal expression annotation systems treated temporal expression recognition as a task along with entity recognition (Krupka & Hausman, 1998), and used simple hand-written rules (Mikheev, Grover, & Moens, 1998). 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.

The recognition task is generally considered to be "do-able" (Ahn, Adafre, & de Rijke, 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.

Temporal expressions are recognised as being highly idiosyncratic, at least in English, but attempts have been made by linguists to make generalisations of the underlying grammar (Flickinger, 1996). Rule-based automated annotators use this principle by attempting to annotate timexes using these rule-based generalisations of the grammar.

Mani & Wilson (2000), in addition to the rule-based tagger discussed below, also experimented with machine learning based systems in order solve the problem of distinguishing between the specific use of the word "today" as a temporal expression and the generic use to mean "nowadays". Following this, a number of machine learning based systems have been developed.

Machine learning systems generally all offer an advantage over other rule-based systems as the tedious creation of rules is avoided, and also allows a certain amount of flexibility between languages. Some rule-based systems (Negri & Marseglia, 2004) maintain that in relatively short periods of time (i.e., one man-month) rule sets can be created which perform adequately. Negri & Marseglia (2004) also suggest that the coverage of rule-based systems can be easily extended by the simple addition of further rules, which can be simpler than improving the performance of machine learning systems.

With performance between machine learning and rule-based systems as close as it is there is no clear superior approach to timex annotation, with different authors extolling the advantages of their chosen approach.

The tasks of automated recognition and normalisation are often rolled into the same tool, although Ahn, Adafre, & de Rijke (2005) argues that separation of these components is beneficial. More recently, larger toolkits handling temporal expressions and relations have emerged (Verhagen, et al., 2005), where each component is modularised.

A number of temporal expression annotators are discussed further below.

2.5.1 TEMPEX AND GUTIME

TempEx (Mani & Wilson, 2000) is a rule-based tagger that accepts a document tokenised into words and sentences and tagged for part-of-speech. A number of operations are applied to this input document, the first of which is the identification of the extent of the time expression. A number of regular expression rules are used to define the extent of what should be tagged.

The second module deals with the normalisation of self-contained expressions, and then a third module, called the "discourse processing module", deals with relative

expressions. For relative times, a reference time is established, either from the context of the surrounding sentences or the document creation date, and then rules handle temporal expressions representing offsets from this date by first computing the magnitude of the offset (e.g., "month", "week", etc), and then the direction, either from direct indicators (e.g., "last Thursday") or from the tense of the sentence ("600,000 barrels were loaded on Thursday").

GUTime (Verhagen, et al., 2005) is an extension to the TempEx tagger that extends the capabilities of TempEx to include the new TIMEX3 tag defined in TimeML, as well as some expressions not handled by TempEx, such as durations, some temporal modifiers, and European date formats.

When evaluated against the TERN data, GUTime scored an f-measure of 0.85 and 0.82 for TIMEX2 recognition and normalisation respectively (Verhagen, et al., 2005).

The GUTime program itself has a number of deficiencies which make extending this software difficult. The tagging aspects of TempEx are provided in a number of very large Perl functions which are driven by a Perl script. This is wrapped around by another Perl script and additional rules were added to the TempEx Perl module to create GUTime.

When incorporated into toolkits, such as TARSQI (Verhagen, et al., 2005), there is yet again another wrapper to fit this into the toolkit. These multiple levels of wrappers are code which hides issues due to the monolithic nature of the core TempEx code. In particular, there is a very heavy coupling between the higher level tagging logic and the actual tagging rules – a single function is used which contains all the rules and logic. Similarly, the second and third modules as outlined above are coupled together into a single function.

This program structure makes adding or changing rules difficult due to the coupling between the rules and the logic itself, and makes analysis of the rules difficult.

2.5.2 CHRONOS

Chronos (Negri & Marseglia, 2004) was a system created for the 2004 TERN evaluation that, like GUTime, provides one system for recognition and normalisation. However, these two tasks are split into separate internal components. Chronos is designed to be a multi-lingual system, coping with both English and Italian text.

One main difference between Chronos and GUTime is that Chronos can handle plaintext; tokenisation and part-of-speech tagging occurs in the first phase of the program. This does have the downside of making Chronos more difficult to componentise; if it were to be incorporated into a larger system, this pre-processing may want to be separated out to use a better system.

The recognition phase of Chronos uses about 1000 hand-written rules (considerably more than GUTime), which not only identify an expression and it's extent, but are also used to collect information about an identified expression (such as modifiers and other "clues") which help the later normalisation phase. Additional rules also exist which

handle conflicts between possible multiple tagging. In GUTime, this is handled by an implicit rule ordering.

Additionally, Chronos, in contrast to GUTime which has a clear separation of components, appears to have a heavier coupling and a more integrated system. This recognition phase results in an intermediate representation – an extension of the TIMEX2 standard – which provides the metadata detected in the recognition phase as additional attributes to a tag.

Although this intermediate representation causes a heavy coupling between the two modules of Chronos, it may offer some advantages in reducing any repetition between the two modules by utilising all the information gleamed in the recognition stage.

Normalisation continues in a similar way to that proposed by (Mani & Wilson, 2000). Expressions are classified as either being absolute or relative, and then in the case of relative dates, the direction and magnitude of the relativity is determined and combined with a base date (determined in the recognition phase) to produce an anchor in time.

At TERN 2004, Chronos achieved the best results, with an f-measure of 0.926 and 0.872 for recognition and normalisation respectively, a performance which the authors put down to their more extensive rule set.

2.5.3 DANTE

DANTE (Mazur & Dale, 2007) was a system submitted for the later 2007 TERN evaluation, again using the TIMEX2 schema.

Like Chronos, DANTE takes in plain text, so suffers from the same issue of componentisation as Chronos. Also similar to GUTime and Chronos, DANTE uses grammar rules (using the JAPE system) for identification of timexes. In this recognition phase, a "local semantic encoding" is used, which is an extension of the TIMEX2 standard that produces a (typically underspecified) value for the TIMEX2 'val' attribute. The interpretation phase then takes this "local semantic encoding" and transforms it into a document-level encoding, using a number of assumptions on the progression of the timeline through the document.

Despite the different thought processes behind this (considering the semantics of a timex), the actual system is very similar at a high level to Chronos, yet an F-measure of only 0.7589 was achieved for TIMEX2 extent recognition, so performance is lower.

2.5.4 ATEL

ATEL (Hacioglu, Chen, & Douglas, 2005) differs from the systems presented to this point in that it uses a machine learning approach to recognition, however does not handle normalisation at all.

ATEL takes full advantage of the machine learning approach to flexibility with languages by testing both Chinese and English, but different feature sets are required for both languages, and the results between them differ. The system scans for words as tokens, and then classifies each token as either 'O' for outside a time expression, or '(*', '*', or '*)'

for the beginning, inside or end of a time expression respectively. The input to the system is already expected to be segmented into sentences and tokenised in order to facilitate this.

Each word is associated with a number of features in a sliding window, and a support vector machine classifier is used to classify tokens, expanding the possible classifications to include classes like '((*' and '*))' to allow for embedded expressions.

At the 2004 TERN evaluation, the system scored an f-measure of 0.935 and 0.905 for TIMEX2 detection in English and Chinese respectively.

2.5.5 TIMEXTAG

TimexTag (Ahn, Rantwijk, & de Rijke, 2007) uses a machine learning approach, but unlike ATEL, also incorporates normalisation. Unlike the rule-based systems covered, TimexTag contains two distinct components for recognition and normalisation and concentrates on maximising performance of each component, rather than as an overall system.

Unlike ATEL, TimexTag does not identify timex phrases by considering the individual tokens, but treats it as a phrase classification task, by classifying each node in a parse tree as timex or non-timex. Again, support vector machines are used with a number of lexical and parse-based features.

Once these timexes have been identified, a classifier is used to categorise the phrases into the type of timex they represent semantically (e.g., recurrence, duration, a point in time, and the vagueness of these). Once again, a SVM is used for this classification, and the same features as in phrasal identification are used.

The TimexTag system isn't based completely on machine learning, as rules are used to compute an under-specified representation for the start of the normalisation phase. However, these rules number considerably fewer than in other systems (89 vs. the 1000+ in Chronos). As with other systems, a base date, or "temporal anchor" is used to compute relative dates, and this is determined using simple heuristics. As with other systems discussed previously, the magnitude and direction of a relative timex also needs to be determined, which in TimexTag is once again using a SVM, utilising the same feature sets as before, but also considers tense of surrounding verbs as a feature (a similar approach to Mani & Wilson, 2000).

At the 2004 TERN evaluation, an f-measure of 0.899 was scored, although the absolute f-measure was lower.

2.5.6 RULE INDUCTION

An alternate machine learning approach to temporal annotation is that of rule induction. Baldwin (2002) presented a language-independent temporal expression annotation scheme that uses rule induction techniques to generate rules from an annotated corpus.

The rule induction method implemented here first attempts to classify the incoming TIMEX tags into types (durations, references, dates, and set-denoting expressions) and

specificity (absolute/fully specified, relative/underspecified, and containing 'X' placeholders). Fully specified data is then processed separately, in order to discover a standard form for natural language expressions of dates which can be used with less specified expressions. The learning component then creates a regular expression with which to match the rule, and a set of instructions with which to evaluate the value.

This system obtained an f-measure score of 0.220 for recognition and 0.091 for normalisation, but this was against the French dataset, not the TERN dataset, so is not directly comparable to the other systems presented here.

Later work (Jang, Baldwin, & Mani, 2004) built on this with Korean text. Here, morphological analysis and a stop list are used to match temporal expressions in a text from a dictionary. Extending the annotator to include part-of-speech information and information about temporal modifiers is identified as a technique to automatically build this dictionary. Normalisation of temporal expressions is instead based on a rote-learning technique, where memorisation of relative expressions and their relative values is used, instead of attempting to generalise these as in Baldwin (2002). The scores here were considerably better, with an f-measure of 0.869 for normalisation against the Korean corpus.

3 PROJECT AIMS

Considering the survey of the literature above and the original defined scope of the project, the following project aims can be defined:

- The creation of a Python tool for normalisation and recognition of temporal expressions;
- The implementation of a rule engine and rule set for this tool, taken from the GUTime rule set;
- Investigation into machine learning techniques that can be used for normalisation and recognition with a speculative implementation of these techniques into the tagger.

Further, requirements for the new Python tool can be set:

- To use good software engineering principles to make a tool that can be maintained to incorporate new techniques and research;
- To provide the bulk of the code as a library which can be easily incorporated into toolkits if required, with a simple frontend to this library;
- Following the recommendations given in Ahn, Adafre, & de Rijke (2005), to decouple and modularise the recognition and normalisation components;
- A rule-based engine which allows for the separation of the rule engine from the rules;
- A machine learning engine which can be substituted in place of the rule-based engine;
- Separation of output format from logic in order to avoid the issues seen in previous taggers where a single format was targeted. The tool should be able to output both TIMEX2 and TIMEX3 tags.

It is also useful to define a limit to the scope of this system. Only annotation of the TIMEX tag is considered here, other tags for temporal relations, events, etc, are not considered in this project.

4 WORK PLAN

In order to implement the project aims as defined above, then the work needed to be done can be broken down into a number of components. The first is to implement the core tagger and rules, the second is to investigate and implement a machine learning component into the system, and the final is to evaluate the tagger performance compared to other systems.

Below, more in-depth plans for the system are outlined, followed by a Gantt chart of the identified tasks and estimated workload. The estimated workload errs on the side of caution in order to allow for slippages, and additionally, weekends are not included in the time planning which allows for additional slack to be utilised if needed.

A final point to be made is that of the actual writing of the dissertation. There is no single "write up" task identified below, instead the actual writing of the dissertation is incorporated into the various tasks.

4.1 PYTHON TAGGER

In order to keep with good software engineering practices, the tagger will use source control and unit tests in order to help ensure high quality code. The first task required for this system will therefore be to set up a source control repository and an appropriate continuous integration system for unit tests. Following this, work on the rule engine can commence, using a test-driven development style. In order to aid development, a front end to the system

Once the rule engine has been created, the rule sets for both the recognition and normalisation components need to be created, which will be done by using the rules in the GUTime software (Verhagen, et al., 2005).

ID	Task Name	Estimated Workload
1	Set up of source control and continuous integration	1 day
2	Implementation of rule engine	1 week
3	Front end for tagger	3 days
4	Creation of rule set for recognition	2 weeks
5	Creation of rule set for normalisation	2 weeks

4.2 MACHINE LEARNING TECHNIQUES

Following the implementation of the basic tagger, an investigation into machine learning based techniques that could be implemented as modules for the tagging system. This work will be experimental and implemented building on previous work (as discussed in section 2.5) as inspiration.

ID	Task Name	Estimated Workload
6	Implementation of an SVM approach	2 weeks
7	Implementation of a rule learning approach	2 weeks

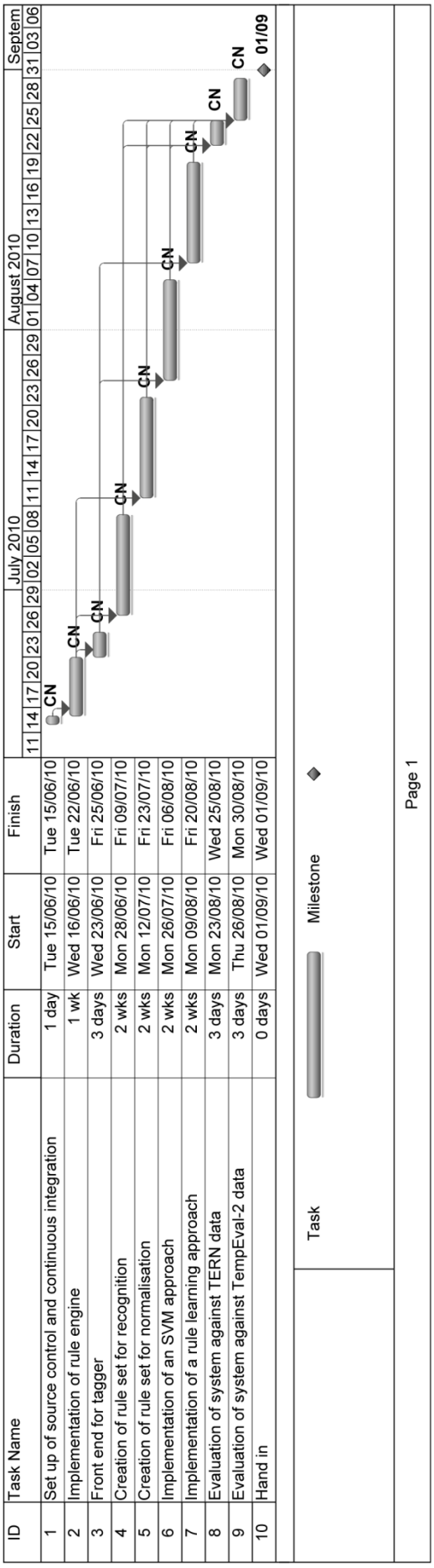
4.3 EVALUATION

In order to evaluate the final tool, then evaluation of different component combinations will be performed (e.g., rule-based recogniser and machine learning based normaliser, etc) in order to discover the best combination and therefore the overall headline performance for the system. The metrics used for evaluation will be the same as used in the TERN contest – f-measures of recognition, extent and normalisation (discussed in section 2.3). The TERN contests provided software for scoring which can be re-used here. This would allow for comparative results between TERNIP and other systems at the TERN contest.

Additionally, the most recent contest for handling temporal expressions, TempEval-2 (Pustejovsky & Verhagen, 2009), included a subtask which specifically dealt with recognition and normalisation of temporal expressions. Evaluation against the TempEval contest dataset would allow for an evaluation of the software against a more up-to-data dataset and for comparison against other state of the art systems.

ID	Task Name	Estimated Workload
8	Evaluation of system against TERN data	3 days
9	Evaluation of system against TempEval-2 data	3 days

4.4 TIMESCALE



5 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.

Baldwin, J. (2002). *Learning temporal annotation of French news.* Washington, DC: Graduate School of Arts and Sciences, Georgetown University.

Ferro, L., Mani, I., Sundheim, B., & Wilson, G. (2001). *TIDES Temporal Annotation Guidelines.* MITRE.

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

Hacioglu, K., Chen, Y., & Douglas, B. (2005). Automatic Time Expression Labeling for English and Chinese Text. *Proceedings of the 6th International Conference in Computational Linguistics and Intelligent Text Processing,* (pp. 548-559). Mexico City, Mexico: Springer.

IDEAlliance. (2008). *Publishing Requirements for Industry Standard Metadata.* IDEAlliance.

Jang, S. B., Baldwin, J., & Mani, I. (2004). Automatic TIMEX2 tagging of Korean news. *ACM Transactions on Asian Language Information Processing (TALIP) , 3* (1), 51-65.

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.

Mazur, P., & Dale, R. (2007). The DANTE Temporal Expression Tagger. In *Human Language Technology. Challenges of the Information Society: Third Language and Technology Conference, LTC 2007, Poznan, Poland, October 5-7, 2007, Revised Selected Papers* (pp. 245-257). Springer-Verlag.

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

Negri, M., & Marseglia, L. (2004). Recognition and Normalization of Time Expressions: ITC-irst at TERN 2004. *TERN 2004 Evaluation Workshop.*

Pustejovsky, J., & Verhagen, M. (2009). SemEval-2010 task 13: evaluating events, time expressions, and temporal relations (TempEval-2). *DEW '09: Proceedings of the Workshop on Semantic Evaluations: Recent Achievements and Future Directions* (pp. 112-116). Boulder, Colorado: Association for Computational Linguistics.

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* .

Pustejovsky, J., Verhagen, M., Sauri, R., Littman, J., Gaizauskas, R., Katz, G., et al. (2006, April 17). TimeBank 1.2. Philadelphia: Linguistic Data Consortium.

Saurí, R., Littman, J., Knippen, B., Gaizauskas, R., Setzer, A., & Pustejovsky, J. (2006). *TimeML Annotation Guidelines Version 1.2.1.*

Setzer, A. (2001). *Temporal Information in Newswire Articles: An Annotation Scheme and Corpus Study.* PhD dissertation, University of Sheffield.

Setzer, A., & Gaizauskas, R. (2001). A pilot study on annotating temporal relations in text. *Proceedings of the workshop on Temporal and spatial information processing* (pp. 1-8). Association for Computational Linguistics.

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., & Moszkowicz, J. (2008, January). AQUAINT TimeML 1.0 Corpus. 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.