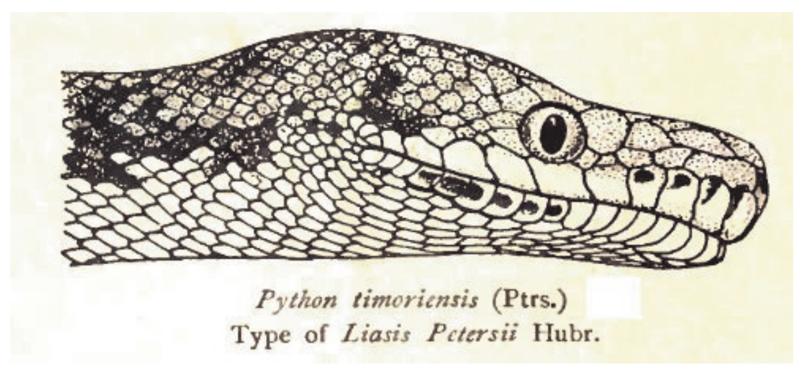
TERNIP Temporal Expression Recognition and Normalisation in Python

Christopher Northwood, MSc Computer Science with Speech and Language Processing

Supervised by Dr Mark Hepple

Department of Computer Science, The University of Sheffield



Abstract

This dissertation presents TERNIP (Temporal Expression Recognition and Normalisation in Python), a system for recognition of temporal expressions in text and normalisation of those expressions to a concrete date and time. TERNIP is modular and agnostic to output format, supporting both TIMEX2 and TimeML standards. Recognition and normalisation is implemented using a rule engine and rule set converted from the GUTime tool, which scores an f-measure for recognition of 0.68 and 0.82 for normalisation against the TERN corpus, comparable to the performance of GUTime. This modular nature of TERNIP encourages the creation of future robust annotation modules.

A Quick Introduction

Temporal expressions are words and phrases which refer to some point in time, e.g., “Shall we meet next Wednesday?”, ‘next Wednesday’ is a temporal expression.

Recognition is the task of identifying these temporal expressions in a block of text.

Normalisation is the task of resolving these expressions to a particular point in time.

TIMEX2 and **TimeML** are two standards for annotating temporal expressions in XML.

TERN and **TempEval-2** are two contests that measure performance of systems in recognition and normalisation

GUTime is a recognition and normalisation tool created by Mani & Wilson that uses regular expressions and rules

Performance

TERN evaluation	**TERNIP**	**GUTime**
Recognition (any overlap)	0.68	0.68
Recognition (extents match)	0.57	0.56
Normalisation (`val` attribute)	0.82	0.57
TempEval-2 evaluation		
Recognition (per-token)	0.78	0.75
Normalisation (`value` attribute)	0.69	0.65

F1-measure of recognition of temporal expressions and normalisation of these values

Sample TimeML

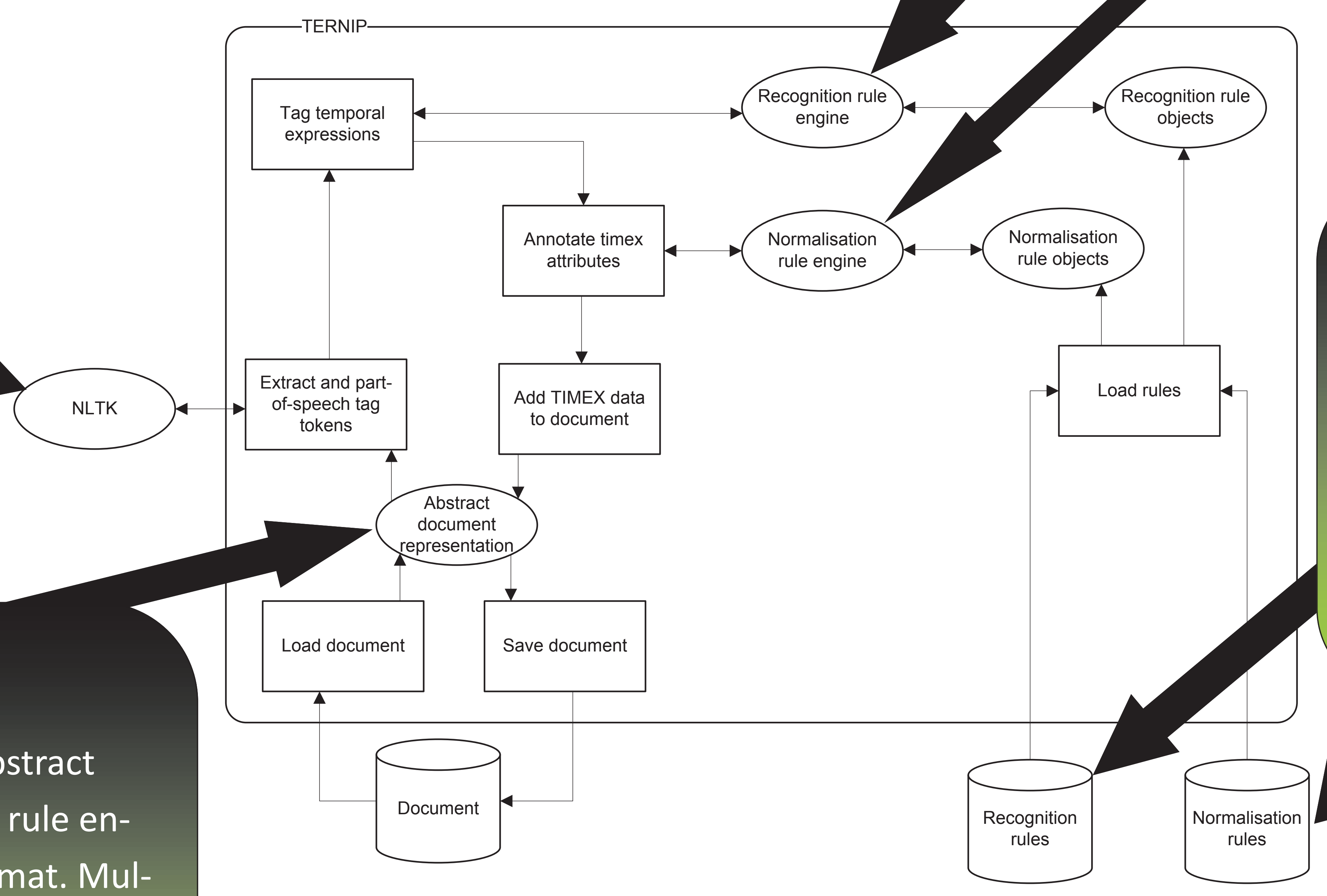
```
INDEPENDENCE, Mo. _ The North Atlantic Treaty Organization
embraced three of its former rivals, the Czech Republic,
Hungary and Poland on <TIMEX3 tid="t3" type="DATE" val-
ue="1999-03-12">Friday</TIMEX3>, formally ending the Soviet
domination of those nations that began after World War II
and opening a new path for the military alliance.
```

Rule Engines

The rule engine is responsible for loading the rules from disk, ordering the rules based on the dependencies specified in the rules and then executing these rules

NLTK

The NLTK is used to split the document into sentences, then tokenise and part-of-speech tag input documents


TERNIP
Tag temporal expressions
Recognition rule engine
Recognition rule objects
Annotate timex attributes
Normalisation rule engine
Normalisation rule objects
NLTK
Extract and part-of-speech tag tokens
Add TIMEX data to document
Load rules
Abstract document representation
Load document
Save document
Document
Recognition rules
Normalisation rules

Rules

Rules are represented as a formalism in text files, specifying conditions for execution, and the action to be taken when these conditions are matched. The exact form varies between recognition and normalisation rules

Documents

Documents are loaded into an abstract representation, which allows the rule engines to work on a consistent format. Multiple formats are supported, which can convert documents into an internal format, and then add annotation details back to the document

Sample Recognition Rule

```
Type: time
Match: (<(about|around|some)~.+>)?<(noon|midnight|mid-?day)~.+>
```

This rule will recognise expressions such as “around noon” or “about midnight” using regular expressions and create a timex tag of type ‘time’. The angular brackets indicate token boundaries and the ~ indicates a separator between the token and its part-of-speech tag.