10.12871/clicit201426

Experiments in Identification of Italian Temporal Expressions

Giuseppe Attardi
Dipartimento di Informatica
Università di Pisa
Largo B. Pontecorvo, 3
I-56127 Pisa, Italy
attardi@di.unipi.it

Luca Baronti
Dipartimento di Informatica
Università di Pisa
Largo B. Pontecorvo, 3
I-56127 Pisa, Italy
lbaronti@gmail.com

Abstract

English. We describe our experiments in participating to the EValuation of Events aNd Temporal Information (EVENTI) task, for the EVALITA 2014 evaluation campaign. We used the HeidelTime tagger extended with a wrapper for the Tanl POS tagger and tokenizer of the Tanl suite. The rules for recognizing Italian temporal expressions were rewritten and extended after the submission, leading to a 10 point increase in F1 over the Italian rules in the HeidelTime distribution.

Italiano. *Nell'articolo descriviamo gli esperimenti svolti per la nostra partecipazione al task EValuation of Events aNd Temporal Information (EVENTI), nel'ambito della campagna di valutaionze EVALITA 2014. Per il riconoscimento e normalizzazione delle espresioni temporali abbiamo utilizzato il tagger HeidelTime, estendendolo con un wrapper per poter utilizzare il POS tagger e il tokenizer della suite di NLP Tanl. Le regole per il riconoscimento delle espressioni temporali in italiano sono state riscritte ed estese, dopo la sottomissione, ottenendo un miglioramento di 10 punti di F1 rispetto alle regole presenti nella distribuzione di HeidelTime.*

1 Introduction

The shared task EVENTI at Evalita 2014, required to recognize temporal expressions within a corpus of Italian text documents and to normalize them according to the It-TimeML TIMEX3 specifications.

Training and test data are distributed in the CAT (*Content Annotation Tool*) (Bartalesi Lenzi et al., 2012) labelled format. This is an XMLbased standoff format where different annotation layers are stored in separate document sections and are related to each other and to source data through pointers.

2 Approach

We chose to use an available temporal tagger and to adapt it for the task. HeidelTime (2014) is a cross-domain temporal tagger developed at the Database Systems Research Group at Heidelberg University (Strötgen and Gertz, 2013). For detecting temporal expressions, HeidelTime uses a set of rules based on regular expressions and conditions on the POS tags of words matched by those expressions. The rules also contain normalization directives for producing the TIMEX3 notation.

HeidelTime provides a plugin architecture, relying on external tools for performing tokenization and POS tagging. The current distribution provides wrappers for TreeTagger (Schmid, 1994), Stanford POSTagger and JVnTextPro.

The standalone version of HeidelTime requires a plain text as input and returns a TimeML (Pustejovsky et al., 2003) document containing the original text with the temporal expressions enclosed within a TIMEX3 element.

HeidelTime is based on the UIMA architecture, that orchestrates the processing of data among CAS processors, passing CAS objects from one stage to the next forming a pipeline. Typically the HeidelTime pipeline consists in three stages: tokenization, POS tagging and sen-

tence annotation. The first two stages are delegated to wrappers for external tools, the third one is dealt by HeidelTime itself. Those tools that provide a UIMA interface are called directly in memory; the others are invoked through wrappers that pass them as input a plain text file and collect the annotations to be added to the CAS from their output. This is the case for TreeTagger.

In our case, we wished to use the tools from Tanl (Text Analytics in Natural Language) (Attardi, Dei Rossi and Simi, 2010) a suite of statistical machine learning tools for text analytics based on the software architecture paradigm of data pipelines. Differently from UIMA, where each stage must process a whole document before it can be handled to the next stage, in a data pipeline processing occurs on demand and each stage pulls data as needed from its preceding stage. The granularity of the units of data requested at each stage depends on the requirements of that stage and can vary from a single line of text, a single token or a single sentence.

The Tanl POS tagger (Attardi et al., 2010) is similar to the one that achieved the best results (Attardi et al., 2009) in the task of POS classification at Evalita 2009.

3 Format Conversion

The training corpus is provided in CAT format where text is represented as an ordered list of tokens. The temporal expression information is present in TIMEX3 elements within the Markables element after the tokens. A temporal event in the text is represented by a TIMEX3 element with attributes representing its type and value, and with children elements containing numeric references to the tokens involved.

A special TIMEX3 element with no children is used to store the publication time information[1], useful for the tagger to correctly compute the absolute time for relative[2] temporal expressions like "ieri" or "lo scorso giugno".

A scorer script is provided by the organizers for evaluating the accuracy of a system output. The scorer works with two sets of CAT files, typically the gold annotated reference set and the system output.

We process each document through the following steps:

1. extract the publication/creation date from the document;
2. convert the corpus document to plain text or use the supplied text version of it;
3. invoke HeidelTime supplying both the plain text file and the publication date as parameters;
4. convert the HeidelTime output into CAT format.

Each step, except the 3rd, is performed by a suitable Python script. The whole process is driven by a custom Makefile, in order to automate the process of carrying out or of repeating the experiments.

4 Results

Before the submission deadline we didn't have time to perform any fine tuning of the HeidelTime rules for Italian. Instead, we focused on integrating the Tanl tagger into the HeidelTime pipeline, and to test its out-of-the-box performance. Hence, we didn't exploit the training corpus for tuning or correcting the rules for Italian, and we used a basic model for the Tanl tagger.

We submitted two runs: Unipi_Tanl and Unipi _TreeTagger. Unipi_TreeTagger is a baseline run produced using HeidelTime in its default configuration for Italian, using TreeTagger and the supplied Italian rules. Unipi_Tanl was an attempt to use the tools from Tanl (the Tanl Tokenizer and the Tanl POS tagger) adapting the rules for using the Tanl POS tagset. Unfortunately, as we discovered later delving into the code of HeidelTime, the rules for matching POS tags were written using regular expressions, which turned out not to be supported in the current version of HeidelTime.

This explains why the official scores in Table 1 show no significant difference between the two runs. We corrected this problem after the submission and rewrote the rules for Italian as described in the following section, achieving significant improvements.

POS Tagger	F1 (strict)	F1 (relaxed)
Best	**0.821**	**0.893**
Unipi_Tanl	0.659	0.771
Unipi_TreeTagger	0.662	0.768

Table 1. Results in Task A.

[1] sometimes different from "creation time".

[2] As opposed to an absolute temporal expression like "23 dicembre 1934" which can be correctly tagged without reference to the publication time.

5 Wrapper for TanlTagger

Proper use of the Tanl POS Tagger with HeidelTime requires adding a wrapper for it to the HeidelTime sources.

We wrote a Java class HeidelTimeWrapper, which invokes the Tanl Tokenizer and the Tanl POS Tagger as subprocesses. An even better solution would be to build a CAS processor interface for these tools.

A few changes were required also to the code of HeidelTime itself. In particular for dealing with POS_CONSTRAINT rules, which apply only if the expression belongs to a specified POS class, the original code performed a simple string match between the requested POS and the one in the data. However, the POS tags produced by the Tanl Tagger are more refined than those of TreeTagger and include morphology information. One rule for example involves checking whether a word is a plural noun, but since nouns have both number and gender, it is required to check for either `Smp` (noun, male, plural) or `Sfp` (noun, female, plural). Hence we modified the code to allow specifying constraints by means of regular expressions, so that one could just write `S.p`.

We also had to fix a bug in the code that added an extra empty line and skipped the final newline in the file passed to the tokenizer, which caused misalignments in tokens.

Both these changes were reported to the maintainers of the package and will be included in later releases of HeidelTime.

We also stumbled upon another bug in the rule matching code of HeidelTime: when a pattern contains an alternative like this "`(%reUnit|%reUnitTime)`", where the first alternative is a substring of the second, the second one is discarded.

Furthermore, we discovered another unexplained idiosyncrasy in some pattern behavior that was solved by adding a "\b" in front of them.

6 Error Analysis

In order to analyze the tagger errors, we developed a diff script that compares two CAT documents and lists their differences, i.e. each timex3 present in one and missing from the other and vice versa. The script also signals expressions that are tagged with a different type/value.

On the development set our system achieves these values of accuracy:

	Precision	Recall	F1
strict	0.800	0.809	0.805
relaxed	0.884	0.894	0.889

Table 2. Development results.

The absolute values of the True Positives, False Positives and False Negatives on the training corpus are the following:

	TP	FP	FN
strict	633	149	158
relaxed	699	83	92

Table 3. True and False Positives on the training set.

We investigated the causes of the large number of False Positives. Inspecting the output of our comparison script shows that the errors can be classified into the following types:

- adverbs like `presto`/`subito` or adjectives like `passati`/`futuro` that are excluded in the guidelines
- person ages (`51 anni`)
- double digit numbers (`83`, `86`)
- minor differences, e.g. in the extent of the expression or different time value
- a few legitimate temporal expressions (`una settimana fa`, `mese di settembre`, `notte prima`, `alle 23`, `lunedì`, `prossimo anno`, `ultimo trimestre`).

Further tuning the HeidelTime rules might hence help reducing these errors.

The situation with False Negatives is more complicated. Here is a small sample:

```
91
l'anno
86
data
90
un minimo di cinque
un massimo di quindici anni
l'81
quattro ore tutte le mattine
Verso le 9.3
qualche mese a questa parte
in futuro
ora in avanti
solo mese di settembre
ventiquattr'ore dopo
mese tradizionalmente "caldo"
meno di due anni
oltre un anno
```

A few of these are actually ambiguous (`91`, `86`, `90`, `data`) and would require deeper analysis to

be recognized as years; some are due to problems in HeidelTime rule matching (`l'81`, `qualche mese a questa parte`, `oltre un anno`); others have patterns that could actually be dealt by additional specific rules.

Using the diff script we were able to address several misclassification problems, improving the HeidelTime rule system for Italian. The rules included for Italian in the standard distribution of HeidelTime contained a lot of errors. Many seem due to the fact that the rules appear to be incomplete translations from the Spanish version, as shown in this rule:

```
[Pp]rimera met(àa')
```

which should read instead

```
[Pp]rima met(àa')
```

In order to improve the accuracy we almost completely rewrote the rules for Italian and devoted some effort also in making them more modular, avoiding idiosyncrasies and repetitions.

7 After Submission Results

After revising the Italian rule set, we performed a run on the test set, using the new wrapper for the TanlTagger, achieving a significant accuracy improvement, as reported in Table 4.

POS Tagger	F1 (strict)	F1 (relaxed)
Best	**0.821**	**0.893**
Unipi_Tanl	0.723	0.871

Table 4. After submission results.

8 Conclusions

We explored a rule based approach to identification and normalization of temporal expressions in Italian documents.

We chose to use the HeidelTime kit, which allows developing resources for different languages using a suitable rule syntax.

HeidelTime has already been used in other challenges achieving top results on English documents at the TempEval-2 challenge in 2010.

The rules for Italian provided in the distribution turned out to be fairly poor. By rewriting and extending them we were able to achieve a significant 10 point improvement in F1 relaxed accuracy, reaching a score not far from the best. It should be possible to close the gap with some additional effort. We were slowed down in doing so by stumbling upon some problems in the rule matching algorithm of HeidelTime version 1.7, that are due to be fixed in release 1.8.

In order to better deal with Italian documents, we wrote a wrapper for the Tanl POS tagger, which is reported as one of the best for Italian. The use of POS tags is still fairly limited though: for instance it is used to distinguish whether a four digit number is not a year, by the fact that it is followed by a plural noun. More extensive of rules involving POS constraints might help eliminate some false positives.

An interesting development would be to apply more sophisticated analysis tools, for instance a parser. Compositional meaning representations of temporal expressions could be reconstructed from phrases that contain temporal clues and machine learning could be applied to learn their interpretation as in (Angeli and Uszkoreit, 2013) or (Leey et al., 2014).

References

Gabor Angeli and Jakob Uszkoreit. 2013. Language-Independent Discriminative Parsing of Temporal Expressions. *Proc. of the 51st Annual Meeting of the Association for Computational Linguistics* (ACL 2013).

Giuseppe Attardi and Maria Simi. 2009. Overview of the EVALITA 2009 Part-of-Speech Tagging Task. *Proc. of Workshop Evalita 2009*.

Giuseppe Attardi, Stefano Dei Rossi and Maria Simi. 2010. The Tanl Pipeline. *Proc. of LREC 2010 Workshop on WSPP*, Malta.

Valentina Bartalesi Lenzi, Giovanni Moretti and Rachele Sprugnoli. 2012. CAT: the CELCT Annotation Tool. In *Proceedings of LREC 2012*, 333–338.

HeidelTime. 2014. Version 1.7. Retrieved from http://code.google.com/p/heideltime/

Kenton Leey, Yoav Artziy, Jesse Dodgez, and Luke Zettlemoyer. 2014. Context-dependent Semantic Parsing for Time Expressions.

James Pustejovsky, José M. Castano, Robert Ingria, Roser Sauri, Robert J. Gaizauskas, Andrea Setzer, Graham Katz, Dragomir R. Radev. 2003. TimeML: Robust specification of event and temporal expressions in text. *New Directions in Question Answering*, 28–34.

Helmut Schmid. 1994. Probabilistic Part-of-Speech Tagging Using Decision Trees. *Proceedings of International Conference on New Methods in Language Processing*, Manchester, UK.

Jannik Strötgen and Michael Gertz. 2013. Multilingual and Cross-domain Temporal Tagging. *Language Resources and Evaluation*, number 1, 269–298. Springer.