Temporal Expression Recognition Using Dependency Trees

Paweł Mazur[*]

[*]Institute of Applied Informatics
Wrocław University of Technology
Wyb. Wyspiańskiego 27, 50-370 Wrocław, Poland
Pawel.Mazur@mq.edu.au

Robert Dale[†]

[*,†]Centre for Language Technology
Macquarie University,
NSW 2109, Sydney, Australia
Robert.Dale@mq.edu.au

Abstract

In this paper we present a previously unexplored approach to recognizing the textual extent of temporal expressions. Based on the observation that temporal expressions are syntactic constituents, we use functional dependency relations between tokens in a sentence to determine which words in addition to a trigger word belong to the extent of the expression. This method is particularly attractive for the recognition of expressions with complex syntactic structure, for which state-of-the-art pattern-based taggers are not effective.

1. Introduction

Temporal expressions (TEs) are linguistic expressions that refer to temporal entities (i.e. points or periods in time). The first step in extraction of TEs from texts is the proper recognition of their occurrence. This is generally divided into two phases: **detection** and **extent recognition**. Detection is concerned with 'spotting' the existence of an expression, i.e. finding at least one of its constituent tokens. The goal of extent recognition is to precisely determine where the expression starts and ends, i.e. to establish exactly which tokens make up the expression.

Very often these two tasks are approached using the idea of a **trigger**. This is a lexical item whose presence is a strong indicator that there may be an instance of a temporal expression; for example, names of months, weekdays and temporal units are triggers. Identification of a trigger which is in fact part of a temporal expression means that we have detected a temporal expression. Recognition then extends the span of the hypothesized temporal expression beyond the trigger word itself. Of course, triggers (here underlined) may constitute complete temporal expressions (here italicised) on their own, as in Example (1), but very often, as in Example (2), they are only parts of longer strings.

(1) I did not go to work on *Monday*.
(2) I spent *three and a half months* in Spain.

In the majority of systems presented in the literature the remainder of the extent is determined by means of hand-written recognition grammars.[1] Such an approach requires the development of a number of detailed rules centered on a trigger in the context. This is a laborious task requiring the knowledge engineer to foresee what expressions may appear in texts and to control the interaction between the rules; as the grammar grows in size, it becomes increasingly difficult to provide wider coverage while maintaining a high level of precision. What we need is a more general algorithm which, given the trigger as a seed, can grow the extent outwards to find the complete temporal expression. The TIMEX2 guidelines define TEs in syntactic terms (Ferro et al., 2005, pp. 7, 57–58), and so it seems obvious that such an algorithm might be based on the notion of syntactic constituency; but, surprisingly, we can find no reports in the literature of attempts to use a syntactic parser (performing a deeper analysis than chunking) to determine the extent of a TE beyond the trigger.

In this paper we present our experiments with a novel approach that uses functional dependency relations between the tokens in a sentence to determine which words apart from the trigger belong to the extent of the expression. In our model, we assume that the trigger is the head of a dependency tree corresponding to the temporal expression; so, once we have identified a head, we can easily extract the complete temporal expression provided we can correctly determine the dependency structure of the sentence.

2. Related Work

Most existing TE taggers use rule-based grammars which recognize TEs by matching encoded patterns of specific lexical items; some generalisation may be achieved by using POS tags. While it is the case that these taggers achieve high performance, it takes a lot of effort to provide wide coverage without sacrificing precision; this is an issue that becomes increasingly problematic as the coverage of the grammar grows.

To introduce more generalisation to the rules, Ahn et al. (2005) used shallow (chunk) parsing; this provided them with a sequence of non-overlapping units, which correspond approximately to phrases. Their recognition patterns then looked for chunks headed by trigger words, and combined selected contiguous chunks to form TEs. The results were, however, lower than their corresponding grammar that did not use chunks.

There are also two other approaches to extent recognition that do not involve the identification of a trigger. One, well-known in work on named entity recognition more generally and undertaken in the context of TEs by Hacioglu et al. (2005), is to carry out token classification using the B-I-O model: i.e. each token is classified as being a Border (the first or last) token of a TE, an Inside token (between two border tokens) or being Outside of

[1]Our review of the literature identified 27 temporal expression tagging systems, of which 19 are rule-based.

the expression. Another approach, proposed by Ahn et al. (2007), is to use a machine-learning classifier which, for each candidate phrase in a sentence (e.g., each NP or ADVP), decides whether or not it is a TE. Although the two methods scored relatively highly in evaluations, their disadvantage is that they require significant amounts of training data. We attempt to develop a method which does not need to be trained.

3. The Method

Given the observation that temporal expressions are defined as syntactic constituents of sentences, we might attempt to obtain a higher level of generalisation by using syntactic information. Instead of attempting to predict what sequences of individual words around a trigger might constitute TEs, we can develop a conceptually much simpler algorithm that, provided we have a syntactic analysis of a sentence, would choose which parts of a sentence are TEs. This idea bears some similarities to the approach investigated by Ahn et al. (2007); however, our task is defined quite differently. Whereas Ahn et al. (2007) tried to learn a model of what makes a given constituent a TE, we assume we have the head of a TE given, and try to determine the full extent of the expression.

We can distinguish two popular types of syntactic analysis based on the type of information they output; this may be either a phrase structure which defines the syntactic constituents of a sentence, or a set of functional dependencies between the tokens of a sentence. Given TIMEX2's stipulation that the trigger is the syntactic head of a markable expression (Ferro et al., 2005, p. 7), the choice of a dependency-based approach seems most natural; to determine the full extent of a TE, we only have to extract the dependency subtree which has the trigger as root. Consider Example (3):

(3) He returned some gifts *five days after Christmas.*

Here, *days* is the trigger. The analysis of the sentence provides the following dependencies:[2] days:>five, days:>after, and after:>Christmas. Starting from *days* we traverse the tree and get the full extent of *five days after Christmas.* In principle, this approach should also work for more complex cases, where the temporal expression includes a dependent clause, as in the following example:

(4) I recall *the days when he was the best in the team.*

Here, the dependencies found in the subtree headed by the trigger are days:>the, days:>was, was:>he, was:>when, was:>best, best:>the, was:>in, in:>team, and team:>the.

4. The Experimental Set-up

4.1. The Parsers and Text Tokenisation

In our experiments, carried out using the GATE framework,[3] we use four off-the-shelf parsers: Minipar,[4] Connexor,[5] the Stanford Parser (de Marneffe et al., 2006), and the C&C Parser (Clark and Curran, 2007). We do not retrain the parsers because we do not have enough training data; this is a common scenario in real-life applications.

The Stanford Parser uses a very rich set of 48 dependency categories that result in analyses which violate the common assumption that a dependency analysis should be a tree rather than a graph. Based on an analysis of the categories of the Stanford dependencies and parses of a few example sentences from our data, we decided to omit the following link types from our traversal approach: nsubj (nominal subject), cop (copula), cc (coordination), and conj (conjunct) dependencies.

Minipar treats commas as syntactic nodes that are in functional relations with other tokens; accordingly, we implemented a postprocessing step that shortened the extent of a TE if its last token was a comma.

We used the ANNIE sentence splitter from GATE for all parsers, except for Connexor which carries out its own sentence splitting. Only the C&C Parser expects the text to be already tokenized,[6] while all the other parsers do their own tokenization. Note that Connexor can combine several words into a single token: e.g. *25 December 1956* is a single node in a dependency tree.

	ACE'05 Train	WikiWars
Documents	593	22
Tokens (Alternate Tokeniser)	322k	121k
Sentences (GATE)	18,252	4,869
Sentences (Connexor)	18,843	4,857
TIMEX2	5,428	2,681

Table 1: The comparison of the size of the datasets.

4.2. The Data

For carrying out the experiment we chose two datasets that are publicly available and which contain temporal expressions annotated in accordance with the TIMEX2 standard; these are the ACE 2005 Training corpus[7] and WikiWars (Mazur and Dale, 2010). In Table 1 we compare the sizes of the two datasets in terms of the number of documents, sentences, tokens and temporal expressions they contain.

The ACE corpus contains documents from six different domains: newswire, broadcast news, broadcast and telephone conversations, UseNet discussions, and weblogs. WikiWars contains narratives sourced from Wikipedia; each presents the course of a military conflict.

4.3. The Selection of Triggers

The literature does not provide an agreed-upon or recommended set of temporal expression triggers; even the TIMEX2 annotation guidelines only provide a non-exhaustive list of examples. We therefore needed to derive a trigger list ourselves. We distinguish three types of triggers: word triggers (e.g. month names), four digit numbers (years), and conventionalised alphanumeric patterns (e.g. *17-07-1964*).

[2] We use the notation head:>child to represent a dependency.

[3] See http://gate.ac.uk.

[4] See http://webdocs.cs.ualberta.ca/~lindek/minipar.htm.

[5] See http://www.connexor.eu/technology/machinese/machinesesyntax.

[6] We used the Penn Treebank tokenizer with some some minor modifications (e.g. to not break numbers like *3,000* into separate tokens).

[7] See corpus LDC2006T06 in the LDC catalogue.

```
[Day,]YYYY/MM/DD [HH:MM:SS] [TMZ]
[Day,]DD/MM/YYYY [HH:MM:SS] [TMZ]
HH:MM[:SS] [AMPM] [TMZ]
HH AMPM [TMZ]
HH o'clock [TMZ]
[Day,] YYYY Month DD [HH:MM:SS] [TMZ]
[Day,] DD/Month[/YYYY]
[Day,] Month/DD[/YYYY]
[Day,] [YYYY/]Month/DD
Month/YYYY
```

Figure 1: Conventionalised patterns in the Trigger Tagger.

We analysed the corpora used in the experiments to check the frequencies of the words that appear in the extents of the gold standard annotations; based on this, we obtained a set of 166 word triggers. Our analysis also showed that four digit numbers commonly appear as denoting both years (e.g. *1997*) and hours (e.g. *0545*, referring to 5:45am). However, we limit the range to 1900–2019 to be consistent with Ahn et al. (2005) in this regard.[8]

In Figure 1 we present the conventionalised date and time patterns recognized by rules; elements in square brackets are optional, and the `TMZ` stands for a time difference or a time-zone code (e.g. *GMT*). We also allow full and abbreviated names of months and weekdays to appear within the patterns (`Month` and `Day`, respectively). The implemented grammar is slightly more complex since we allow for some punctuation and ordering variations, and single digit numbers in some positions.

4.4. Evaluation

Performance is measured in terms of **lenient recognition** (detection of the presence of a TE) and **strict recognition** (determination of the full extent of the TE). The goal of the trigger extraction process here is to detect enough TEs to test our syntactic recognition method. For this experiment, the absolute values of the lenient and strict results do not matter; it is the difference between them that is important. The only caveat is that if the lenient scores are low, then this suggests that we may not have captured the full variety of triggers in the data, or we may be generating too many false-positive (spurious) annotations. We therefore need a reasonably broad sample of triggers, so that we do not accidentally miss expressions that have interesting syntax just because we have omitted the corresponding trigger.

The lenient and strict results determine the lower and upper bounds, respectively, for the approach based on functional dependencies. The strict results for recognizing triggers on their own allows us to measure the improvement obtained by using a syntactic parser; the lenient results, on the other hand, provide the upper limit of what we can expect to achieve in the experiment (i.e. the score obtained if every detected expression was also correctly recognized).

In Table 2 we present the results obtained with a tagger which recognizes only the triggers, which we refer to as the Trigger Tagger. We consider the three classes of triggers identified above individually and together. As anticipated, we get the best performance when all three types of triggers are used; for both corpora we get very good coverage in detection (0.89 and 0.88 lenient recall) without compromising the precision too much (0.89 and 0.92), resulting in high lenient F-measures of 0.89 and 0.90.

The relatively high strict recall (0.49 and 0.54) tells us that about half of the TEs appearing in our corpora do not have very complex structure; they can be correctly recognized just with simple word triggers and a quite limited set of rules. This fact should be kept in mind when evaluating any fully-featured tagger. Matching just trigger words yields 0.29 and 0.13 strict recall on our data; the large difference here shows that differences among datasets may be significant.

5. Results

On both corpora the best strict results were obtained using the C&C Parser, and the worst using Minipar. The taggers using the Stanford Parser and Connexor fell in the middle, but the differences here were not large; Connexor was slightly more useful on the ACE corpus, but the Stanford Parser yielded slightly better results on WikiWars. Given space limitations, we only report here the results obtained using the C&C Parser; see Table 3.

We first applied our dependency-based approach to each subset of triggers individually; these are the first three set-ups in Table 3. As expected, the lenient results have not changed at all or only insignificantly compared to recognizing just triggers, as these reflect only the performance for the detection task.[9]

With regard to recognition, the strict F-measure got much higher when using word-triggers, much lower when using conventionalised-pattern-based triggers, and slightly better or worse depending on which dataset we consider for four digit numbers (years).

We note that trigger words on their own yield better strict results on ACE than WikiWars, but after applying the syntactic method, the situation changes: the performance is higher on WikiWars. This means that in ACE there are (proportionally) more expressions which are built just with trigger words, but in WikiWars (where the gain obtained with the method is greater), either there are (proportionally) more expressions headed by word triggers, or the parser performs better; the latter in turn may mean that the sentences are more syntactically well-formed and easier to parse.[10]

Two set-ups, #4 and #5, combine all three sets of triggers. In both set-ups the words are used as the syntactic heads of the expressions, but the date-recognition rules are assumed to detect the full extents of the corresponding TEs (i.e. they are not further extended by the syntax-driven method). The difference is in the use of four-digit

[8] Note that restricting years in this way damages performance on WikiWars, which also contains bare year references to earlier periods.

[9] Growing the extent may impact the scoring of the detection task if the new extents impact the matching of system and gold standard annotations and, in consequence, the number of detected, missing, or spurious expressions.

[10] The ACE corpus contains a number of documents with automatically transcribed speech, weblogs entries and UseNet discussions; these genres are challenging to parse well.

	ACE 2005 Training						WikiWars					
	Strict			Lenient			Strict			Lenient		
Trigger Set	Prec.	Recall	F	Prec.	Recall	F	Prec.	Recall	F	Prec.	Recall	F
Words	0.35	0.29	0.32	0.86	0.71	0.78	0.18	0.13	0.15	0.97	0.72	0.83
Dates	0.75	0.16	0.27	0.94	0.20	0.33	0.94	0.29	0.44	1.00	0.31	0.47
Years	0.17	0.04	0.06	0.96	0.21	0.34	0.34	0.12	0.18	1.00	0.35	0.52
Words+Dates	0.48	0.45	0.46	0.90	0.84	0.87	0.55	0.42	0.48	0.97	0.73	0.83
Words+Years	0.31	0.33	0.32	0.82	0.86	0.84	0.23	0.25	0.24	0.80	0.87	0.84
Dates+Years	0.73	0.20	0.31	0.92	0.25	0.40	0.81	0.41	0.54	1.00	0.50	0.67
Words+Dates+Years	0.49	0.49	0.49	0.89	0.89	0.89	0.56	0.54	0.55	0.92	0.88	0.90

Table 2: The results obtained with different sets of triggers by the Trigger Tagger.

		ACE 2005 Training						WikiWars					
		Strict			Lenient			Strict			Lenient		
Set-up		Prec.	Recall	F	Prec.	Recall	F	Prec.	Recall	F	Prec.	Recall	F
1	(Words)	0.56	0.46	0.50	0.86	0.71	0.78	0.69	0.51	0.59	0.97	0.72	0.83
2	(Dates)	0.33	0.07	0.12	0.94	0.20	0.33	0.82	0.25	0.38	1.00	0.31	0.47
3	(Years)	0.23	0.05	0.08	0.96	0.21	0.34	0.27	0.09	0.14	1.00	0.34	0.51
4	(Words+Year)+Dates	0.65	0.65	0.65	0.89	0.89	0.89	0.73	0.68	0.71	0.92	0.87	0.89
5	(Words)+Year+Dates	0.65	0.65	0.65	0.89	0.89	0.89	0.75	0.71	0.73	0.92	0.88	0.90

Table 3: The results for extent recognition obtained with a dependency-based tagger using the C&C parser.

	ACE 2005 Training						WikiWars					
	Strict			Lenient			Strict			Lenient		
Tagger	Prec.	Recall	F	Prec.	Recall	F	Prec.	Recall	F	Prec.	Recall	F
DepC&C	0.65	0.65	0.65	0.89	0.89	0.89	0.75	0.71	0.73	0.92	0.88	0.90
DANTE	0.75	0.79	0.77	0.88	0.92	0.90	0.93	0.93	0.93	0.98	0.99	0.99

Table 4: The comparison of recognition results of the dependency-based approach with a pattern-based tagger.

numbers; in set-up #4 we attempt to grow extents containing these, but in set-up #5 we treat the numbers as complete TEs in themselves. The difference in the results for these two set-ups is, however, insignificant.

Just as when we used the individual subsets of triggers, the absolute results for WikiWars are higher than those for ACE: 0.73 vs 0.65 F-measure. However, we are more interested in the gain obtained using the syntactic method as compared to recognizing just the triggers, since this shows just how much the dependency trees are useful for extent recognition.

Set-up #1 provides gains which are quite different for the two corpora: 0.44 for WikiWars and 0.18 for ACE (F-measure). If we compare the results from set-up #5 with those obtained when annotating the three sets of triggers as TEs, the gains are very similar for the two corpora: 0.18 for WikiWars and 0.16 for ACE (F-measure). The gain for WikiWars is now much smaller because many temporal expressions are now matched by the rules that recognize dates and times in conventionalised formats; on the other hand, these rules make little difference when processing ACE documents.

We note, however, that there is still a big gap between the strict and lenient results: 0.24 on ACE and 0.17 on WikiWars (F-measure). Also, the absolute performance is lower than that which we can get with a pattern-based tagger; in Table 4 we compare the best results of the dependency-based approach with the DANTE tagger (Mazur and Dale, 2007).

The picture is different if we evaluate the method only on what are called **event-based TEs**. These have complex syntactic structures, as their extent typically contains dependent clauses; consider the following example:

(5) During *the three months between the cease-fire and the French referendum on Algeria*, the OAS unleashed a new terrorist campaign.

For such expressions it seems practically impossible to write a recognition grammar based on matching specific lexical items, parts-of-speech or even syntactic chunks. For a dataset consisting of 122 event-based TEs drawn from WikiWars, our dependency-based approach scored 0.69 strict F-measure, while two pattern-based taggers, DANTE and TERSEO,[11] scored 0.06 and 0.03, respectively.

6. Error Analysis

6.1. Sentence Splitting and Tokenisation

We found that in some cases, e.g. an email presenting a trip itinerary where the text was not organized into sentences, sentence splitting negatively impacted extent recognition.

The tokenisation carried out by parsers sometimes resulted in combining two or more tokens into a single syntactic node. This happens, for example, with date ranges like *12-15 September*, where the two hyphen-separated numbers are treated as a single token, resulting in an extent that does not match the gold standard annotations.

[11] `http://gplsi.dlsi.ua.es/~stela/TERSEO`.

6.2. Parsing Errors

A general problem is the attachment of prepositional phrases and dependent clauses, as shown in Example (6); here, the correct extents are marked with italics and the extents found by the tagger are underlined.

(6) The British Army endured the bloodiest day in its history, suffering 57,470 casualties including ...

In other cases the parses are broken and the extents of the temporal expressions include extra tokens from a phrase following the expression:

(7) ... after stalling for a day against the main resistance line to which the enemy had withdrawn.

These problems might be overcome by re-training the parsers on more similar textual data.

6.3. Heads in Dependencies

There is not always a consensus view on what the role of a given token in a dependency relation should be. The differences between parsers in this regard sometime resulted in a different tree than was expected; in particular, the trigger was not always determined to be the head of a relation.

In some cases, for example in the case of conjunctions, a node can occur as a child in two dependencies, with the result that the output is a directed graph rather than a tree. In an expression like *the spring and early summer of 1943*, this has the consequence that both triggers *spring* and *summer* are extended to cover the entire expression.

6.4. Triggers

Recognizing dates and years as 'fully-extended' TEs (i.e. immune from further extension using the dependency-based approach) also contributed to a number of errors, resulting in a failure to recognize references to decades (*the 1950s*) and modified dates (*early 1950*).[12]

6.5. Heads of Temporal Expressions

It turns out that the assumption that the trigger is the syntactic head of the TE that contains it does not always hold. For example, the head of the expression *the middle of August* is *middle* rather than *August*. The source of this problem lies in an inconsistency in the TIMEX2 guidelines: by their definition of a TE, the expression consists of a trigger and its syntactic pre- and postmodifiers. But in order to capture semantic subtleties (e.g. that only a part of a month is being referred to), the extent is permitted to grow to include additional tokens beyond the syntactic modifiers of the trigger. In WikiWars this affected 14% of all incorrectly recognized expressions.

A similar issue concerns expressions like *the 28th of that month*, where the head is *28th* rather than *month*; however, we did not use ordinal numbers as triggers, and so expressions with this structure were only partially recognized.

[12] However, as we discussed earlier, applying the syntax-based method to these triggers did not improve the overall results, because while some expressions obtain the correct extents, other are damaged by including tokens from outside the correct extent because of parsing errors.

7. Conclusions and Future Work

We have presented a new approach to recognising the extent of temporal expressions in text: given the trigger as a seed, we grow the extent outwards by including all syntactic nodes found in the dependency tree with the trigger as root.

We experimented with four dependency parsers (Minipar, Stanford, C&C and Connexor) using two datasets (the ACE 2005 Training corpus and WikiWars). The best results were achieved using the C&C Parser. The results demonstrate that the method generally works as anticipated. The gap between the performance of the method and the upper bound is, however, significant: 0.24 for the ACE corpus and 0.17 for WikiWars. Our error analysis reveals that in most cases the problem lies in an incorrect syntactic analysis being provided by the parser. We also found that there are TEs where the trigger is not the syntactic head. On WikiWars, this issue is implicated in about 14% of the incorrectly recognized expressions. The source of this problem lies, however, in an inconsistency in the TIMEX2 guidelines.

The method is most useful for the recognition of event-based TEs; these are expressions with complex syntax which cannot be practically recognized by rule-based grammars using lexical and shallow syntactic information such as POS tags or chunking. On a dataset of 122 event-based TEs drawn from WikiWars, the method achieved a strict F-measure of 0.69, significantly outperforming two rule-based systems which scored 0.03 and 0.06.

As future work we plan to experiment with using phrase structures instead of dependency trees, to overcome some of the problems noted above.

8. References

D. Ahn, S. F. Adafre, and M. de Rijke. 2005. Extracting Temporal Information from Open Domain Text: A Comparative Exploration. In *Proc. of the 5th Dutch-Belgian Information Retrieval Workshop*, Delft, The Netherlands, March.

D. Ahn, J. van Rantwijk, and M. de Rijke. 2007. A Cascaded Machine Learning Approach to Interpreting Temporal Expressions. In *Proc. of HLT: The Annual Conference of the North American Chapter of the ACL*, Rochester, NY, USA.

S. Clark and J. R. Curran. 2007. Wide-Coverage Efficient Statistical Parsing with CCG and Log-Linear Models. *Computational Linguistics*, 33(4):493–552, December.

M.-C. de Marneffe, B. MacCartney, and Ch. D. Manning. 2006. Generating Typed Dependency Parses from Phrase Structure Parses. In *Proc. of the IEEE / ACL 2006 Workshop on Spoken Language Technology*.

L. Ferro, L. Gerber, I. Mani, B. Sundheim, and G. Wilson. 2005. TIDES 2005 Standard for the Annotation of Temporal Expressions. Technical report, MITRE, September.

K. Hacioglu, Y. Chen, and B. Douglas. 2005. Automatic Time Expression Labeling for English and Chinese Text. In A. F. Gelbukh, editor, *Computational Linguistics and Intelligent Text Processing, 6th International Conference*, LNCS, pages 548–559. Springer, February.

P. Mazur and R. Dale. 2007. The DANTE Temporal Expression Tagger. In Zygmunt Vetulani, editor, *Proc. of the 3rd Language and Technology Conference*, Poznan, Poland, October.

P. Mazur and R. Dale. 2010. WikiWars: A New Corpus for Research on Temporal Expressions. In *Proc. of the Conference on Empirical Methods in NLP*, pages 913–922.