Identification of time expressions, signals, events and temporal relations in texts

Cyril Perrig
ETH Zurich,
Switzerland,
perrigc@student.ethz.ch

Abstract

In this project an approach to time and event annotation is presented. First time expressions, signals and events are identified in texts. Using these extracted features temporal relations between events are identified. Furthermore the performance of event recognition with increasing corpus size is investigated.
We use the widely accepted *TimeML* specification language as an annotation scheme. The identification tasks are solved by a machine learning technique called *Support Vector Machines (SVMs).*
Finally we report the results we obtained and discuss conclusions and open problems.

1 Introduction

The goal of this project is to detect all the features needed to identify temporal relations in texts and at last to identify these temporal relations. In other words we want to determine a temporal relation between two events. For this we first have to detect time expressions, then signals and finally events. Due to time reasons the focus of this project is set on the detection of events while the identification of temporal relations itself is not fully explored. The results of the event recognition is also compared to a paper from IBM (Boguraev and Ando, 2005).
The order of detection is important because detections of a specific item might be used as a feature input for another detection. Before explaining these identifications in more details a short overview of TimeML is highlighted. Then the machine learning tool used for the tasks and the experimental setup is reviewed. In the next section all tasks are evaluated step by step. Finally the results, conclusions and open problems are discussed.

2 TimeML

TimeML 1.1[1] is a robust specification language for events and temporal expressions in natural language. It is an XML-based language and provides all tags needed for our analysis. Amongst other things it is designed to address the problem of ordering events with respect to one another. Besides all these reasons for choosing TimeML it is widely accepted and used in the language processing research.
In the following subsections all the relevant tags for this project are briefly explained and illustrated with an example. TimeML is by far more complex and therefore the interested reader is referred to the TimeML specifications 1.1 (Sauri et al., 2004) for more details.

2.1 Time Expressions

Time expressions in TimeML are tagged with TIMEX3 tags. This tag is primarily used to mark up explicit time expressions, such as times, dates, durations, etc.
An example TIMEX3-tag (if "today" is the 2006-01-17):

<TIMEX3 type="time" value="2005-01-16T10:00">yesterday at 10 am</TIMEX3>

2.2 Signals

The tag SIGNAL is used to annotate sections of text, typically function words, that indicate how temporal objects are to be related to each other. This can either be an indicator of temporal relations (e.g. *during, when, etc.*) or an indicator of temporal quantification (e.g. *twice, three times, etc.*).

2.3 Events

An EVENT is typically described by a verb, although event nominals, such as "crash" in "...killed by the crash", will also be annotated as events. Further events have a tense, an aspect (both optional), and a *class.* The

[1] http://www.timeml.org

class classifies the type of event, wheter it is e.g. a state or an intentional action. All in all TimeML distinguishes seven different event types. The attributes tense and aspect can each have four different values, *past, present, future, none* and *progressive, perfective, perfective_progressive, none* respectively. In Table 1 an example with tense value *present* is illustrated.

Verb group	Aspect
teaches	*none*
is teaching	*progressive*
has taught	*perfective*
has been teaching	*perfective_progressive*

Table 1: Example with tense value *present*

2.4 Temporal Relations

Temporal relations are marked with TLINK tags. It represents the relation between two temporal elements. Such links can connect time expressions with events or pairs of events. There are thirteen different temporal relation types, e.g. *before, after, includes etc.*
Let's consider the following sentence:
"Peter came home at ten o'clock and after eating, he went to bed."
Obviously there are three events and one time expressions in this example. Between the first event *(came)* and the time expression *(ten o'clock)* the tlink tag is of type *simultaneous.* The second temporal relation connects the two events *eating* and *went to bed* with an after-relation.

2.5 TimeBank

TimeBank 1.1 is an illustration and proof of concept of the TimeML specifications 1.1. It is a set of 186 news report documents annotated with the 1.1 version of the TimeML standard for temporal annotation. We use this freely provided illustration as a corpus in this project. With around 75'000 tokens one has to be aware that this corpus is rather small.
To end this section an example sentence from TimeBank 1.1 containing two events is presented:

On the other hand, it's <EVENT eid="e1" class="OCCURRENCE" >**turning**</EVENT>out to be another <EVENT eid="e84" class="STATE" >**very bad**</EVENT>financial <TIMEX3 tid="t83" type="DURATION" >week</TIMEX3>for <ENAMEX TYPE="LOCATION">Asia</ENAMEX>.

3 YamCha

YamCha[2] (Yet Another Multpurpose CHunk Annotator) is a generic, customizable, open source text chunker oriented towards a lot of NLP tasks. Further it provides a lot of useful features which can easily be used by changing the standard input parameters. If not stated otherwise we use the standard input parameters of YamCha. Concerning the feature sets (window-size) the default setting is illustrated in Figure 1.
YamCha is using a state-of-the-art machine learning algorithm called Support Vector Machines (SVMs). This classification algorithm provides a high generalization performance independent of feature dimension. Combinations of multiple features can be trained by using a Kernel Function. In YamCha only polynomial kernels are supported.

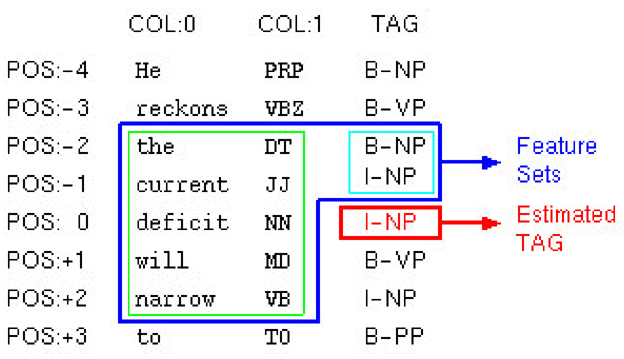

Figure 1: Default feature set (window-size) setting in YamCha

4 Experimental Setup

First of all the corpus has to be tokenized to get a "YamCha-compatible" format. Additionally all relevant information belonging to a token, such as e.g. tense, has to be extracted. Unfortunately the POS (Part-of-Speech) of a token is not provided in the TimeBank-corpus. Therefore we used MXPOST[3] (Maximum Entropy Part-of-Speech Tagger) to get the POS of the tokens.
Another issue we have to be aware of is that sometimes more than one token is inside an entity (or tag). The first token of such an entity is often not very relevant for the detection. An example is illustrated in the time expression

[2]http://chasen.org/ taku/software/yamcha/
[3]http://www.cogsci.ed.ac.uk/j̃amesc/taggers/MXPOST.html

"*the last twenty hours*" where the determiner *the* is certainly not an indicator for a time expression. Therefore we use the IOB-model (Inside Outside Beginning) which distinguishes tokens that begin, are inside or are outside an entity. In Table 2 the end of the previous example sentence is presented.
To evaluate the results for the different tasks the measures *recall, precision and F-measure* are used. In the following subsection a short recall of these measures is given. Furthermore we use a 5-fold cross validation to get an "averaged" F-measure. First the corpus is divided into 5 equally sized parts. Then each part is once used as a test set (20% of corpus) and the others as a training set (80% of corpus). This means that each training set contains around 60'000 tokens. At the end the average F-measure of the 5 runs is taken as a reference measure.

Token	**POS**	**Event**	**TimeExpr**
turning	VBG	B-*occurrence*	O
out	RP	O	O
to	TO	O	O
be	VB	O	O
another	DT	O	O
very	RB	B-*state*	O
bad	JJ	I-*state*	O
financial	JJ	O	O
week	NN	O	B-*timex3*
for	IN	O	O
Asia	NNP	O	O
.	.	O	O

Table 2: Example illustrating the IOB-model for the features *POS, Event and Time Expressions*

4.1 F-measure

As mentioned before the measures recall, precision and F-measure are used. They are common when evaluating performance of algorithms in computational linguistics. The definitions can easily be explained with the Figure 2 where the sets A, B and C are shown. Recall measures how many of the relevant entities that were found. It is defined as *Relevant items retrieved / All relevant items* = $\frac{B}{A+B}$. Precision is a measure of how many of the retrieved entities are relevant. It is defined as *Relevant items retrieved / All retrieved items* = $\frac{B}{B+C}$. Recall (R) and precision (P) are combined into the F-measure which is the harmonic mean of both numbers:

$$F = \frac{2 * P * R}{P + R}$$

For more information about these measures the reader is referred to (Nugues, 2005).

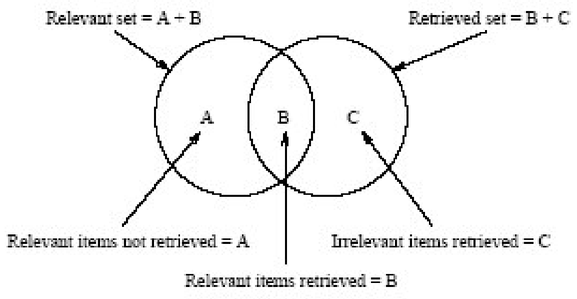
Relevant set = A + B
Retrieved set = B + C
A
B
C
Relevant items not retrieved = A
Irrelevant items retrieved = C
Relevant items retrieved = B

Figure 2: Precision and Recall

5 Experiments

5.1 Time Expressions

Extracting time expressions from texts comes at the first place of our analysis tasks. The only features used for training are the token itself, the POS and of course the time expression. In Table 3 the results for the 5 runs are listed. The average F-measure is 0.791 which is quite high considering that only approximately 2'200 time expressions are contained in each training set. This means that around 3.7% of all tokens of each training set are tagged as time expressions.

Recall	**Precision**	**F-measure**
0.590566	0.894286	0.711364
0.718033	0.914405	0.804408
0.723842	0.917391	0.809204
0.791878	0.921260	0.851683
0.770355	0.783439	0.776842

Table 3: Results for Time Expressions

5.2 Signals

For identifying Signals we even have less information in the training sets (1'800 Signals per set). As we already extracted time expressions these entities are now of use for the next task. Nevertheless we take all features used for a specific task from TimeBank and not from an output of our own detection tasks. This fact holds

throughout this project.
The experiment using only the token itself and the POS results in a F-measure of 0.576. Adding the time expressions to the features improves the F-measure to 0.616. One reason for this moderate result is certainly the small corpus size.

5.3 Events

As noted before TimeML provides 7 different class types for events. Obviously this makes this analysis task very difficult. On this account we evaluate the experiment for two different cases. Once events are recognized without considering the class type *(untyped case)* and once we want to determine the events together with their class type *(typed case)*. Each training set contains approximately 7'300 events with unequally distributed class types.
In Table 4 the F-measure for 7 different experiments are presented. The first 3 rows differ only in the feature set. In the last row the window size parameter is increased while using the same features as in the third row. This experiment is only evaluated for the untyped case because of the long run-time for training. Further the untyped case shows that increasing the window size doesn't improve the performance. The evaluation of the typed case using the features token, POS, TimeExpr and Signal takes around one hour per run on an Intel Mobile Pentium 4 with 1.7 GHz. Thus the real run-time for an experiment, using a 5-fold cross validation, is around 5 hours.
As expected the best results are obtained using the features token, POS, TimeExpr and Signal together with the standard settings in YamCha. These results are only around 3% lower than the results from (Boguraev and Ando, 2005). Though their feature vector representation is much more complex and contains a lot more features, e.g. word uni- and bi-grams based on subject-verb-object and preposition-noun constructions. Compared to our SVM-approach they use a classification framework based on a principle of *empirical risk minimization* called *Robust Risk Minimization* (RRM).
Finally we investigated the performance of event identification with increasing corpus size. In the first step the training set size is only 20% of the corpus and therefore of the same size as the test set. Then the training set size is increased until it reaches 80% of the corpus as in the experiments before. Because of run-time reasons we disclaim the 5-fold cross validation and only evaluate one run per training set size. In Figure 3 and 4 the F-measure is plotted against the percentage of the corpus size for both, the untyped and the typed, case. The irregularity in Figure 4 is probably caused by the fact that we do not use 5-fold cross validation. Apart from this and that the F-measure is higher in the untyped case the two plots illustrate the same logarithmic trend. They also show that the point where the performance begins to stagnate is not yet reached. Therefore a larger corpus can still improve the performance significantly.

	typed	untyped
Token/POS	0.551	0.734
Token/POS/TimeExpr	0.557	0.735
Token/POS/TimeExpr/Signal	0.573	0.754
larger window size	-	0.751

Table 4: Results for Events

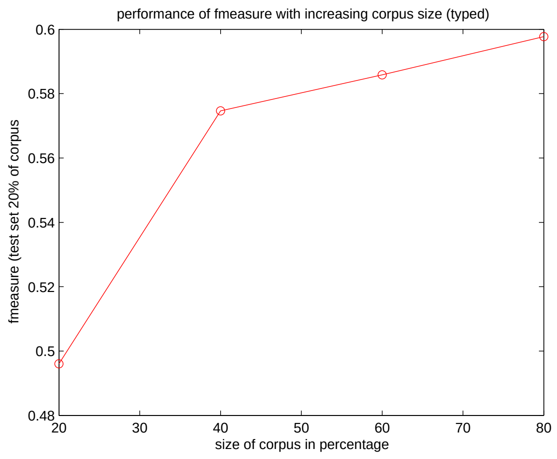

Figure 3: Increasing corpus size (typed case)

5.4 Temporal Relations

Determining temporal relations is even a more complex problem than event recognition. In fact TimeML provides 13 different temporal relation types. To simplify the problem we focus only on six temporal relation types (*before, after, includes, is_included, simultaneous and identity*). Furthermore we only consider temporal relation between events.
We use a similar approach to the one in (Berglund, 2004) that builds a feature row considering two adjacent events. All in all 15 features are used for the temporal relations

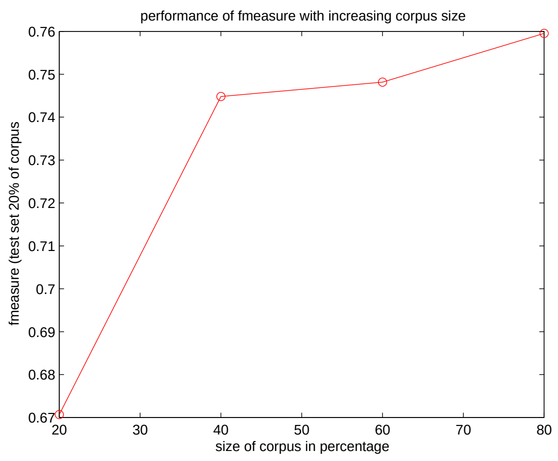

Figure 4: Increasing corpus size (untyped case)

analysis. In the following listing the features are presented:

- 1. - 5. Token/POS/event/tense/aspect of first event
- 6. - 10. Token/POS/event/tense/aspect of second event
- 11. temporal signals between events: [several, none, signal_token]
- 12. distance measured in tokens: [1, 2t3, 4t6, 7t10, gt10]
- 13. distance measured in sentences: [0, 1, . . . , gt4]
- 14. distance measured in punctuation signs: [0, 1, . . . , gt5]
- 15. temporal relation

This feature representation is illustrated on the following example:
The cat ate some cheese. Then, the dog saw the cat and chased it. Cheese is good for you.
Three events (e1:ate, e2:saw and e3:good) and one signal (signal:then) are contained in this example. This gives us three pairs of adjacent events with the following representation:

1. ate VBD OCCURRENCE PAST NONE saw VBD OCCURRENCE PAST NONE THEN 7t10 1 2 BEFORE
2. saw VBD OCCURRENCE PAST NONE chased VBD OCCURRENCE PAST NONE NONE 4t6 0 0 BEFORE
3. chased VBD OCCURRENCE PAST NONE good JJ STATE NONE NONE NONE 4t6 1 1 NONE

If we only consider adjacent events each training set size contains around 6'650 pairs of events where around 1'220 pairs possess a temporal relation. To get more pairs of events *i_before and i_after (i for immediately)* are mapped to the temporal relations *before and after*.
The results of the 5 runs are listed in Table 5 which gives us an average F-measure of 0.195. The five runs show a quite poor performance (especially the recall is very low) and a high variance.

Recall	**Precision**	**F-measure**
0.087227	0.237288	0.127563
0.130159	0.284722	0.178649
0.219355	0.441558	0.293103
0.16	0.360902	0.221709
0.134752	0.183575	0.155419

Table 5: Results for Temporal Relations

6 Conclusions and Open Problems

We showed that SVM is a very powerful algorithm for this kind of text analysis. With a much easier feature representation we could almost reproduce the results from (Boguraev and Ando, 2005).
The main bottleneck could easily been located with the very small corpus size. Another problem is that the entity types are often unequally distributed, e.g. the TimeBank 1.1 contains 4'452 events of type *occurrence* and only 51 of type *perception*. In the future better results could definitively be reached with a much larger corpus.
In the last case of event ordering the results are quite poor. The task of identifying temporal relations is known to be very difficult, especially if it is about domain-independent text. But the results could be easily improved by creating a larger corpus containing more event pairs. This can be realized by considering a larger event window size and not only considering adjacent events. The best strategy is to take the transitive closure to build event pairs. The number of related pairs increases and therefore the training sets provide more training information.
Another approach to improve the results could be realized by tuning the parameters for the SVMs. Although in an experiment for the event recognition task using the 3nd degree of polynomial kernel no improvement could be observed.

7 Acknowledgements

First of all I would like to thank my supervisor Richard Johansson for his patience, support and good advices. Additionally, I would like to thank Pierre Nugues for giving me the opportunity to work on this project.

References

A. Berglund. 2004. Extracting temporal information and ordering events for swedish.

B. Boguraev and R.K. Ando. 2005. Timeml-compliant text analysis for temporal reasoning. *IJCAI-05*, pages 997–1003.

P. Nugues. 2005. *An Introduction to Language Processing with Perl and Prolog.*

R. Sauri, J. Littmann, R. Gaizauskas, A. Setzer, and J. Pustejovsky. 2004. Timeml annotation guidelines, version 1.1.