UZurich in the BioNLP 2009 Shared Task

Kaarel Kaljurand
Institute of
Computational Linguistics
University of Zurich
Switzerland
kalju@cl.uzh.ch

Gerold Schneider
Institute of
Computational Linguistics
University of Zurich
Switzerland
gschneid@cl.uzh.ch

Fabio Rinaldi[*]
Institute of
Computational Linguistics
University of Zurich
Switzerland
rinaldi@cl.uzh.ch

Abstract

We describe a biological event detection method implemented for the BioNLP 2009 Shared Task 1. The method relies entirely on the chunk and syntactic dependency relations provided by a general NLP pipeline which was not adapted in any way for the purposes of the shared task. The method maps the syntactic relations to event structures while being guided by the probabilities of the syntactic features of events which were automatically learned from the training data. Our method achieved a recall of 26% and a precision of 44% in the official test run, under "strict equality" of events.

1 Introduction

This paper describes the adaptation of an existing text mining system to the BioNLP shared task. The system has been originally created for participation in the BioCreative[1] protein-protein interaction task (Rinaldi et al., 2008) and further developed for an internal project based on the IntAct dataset of protein interactions (Kerrien et al., 2006). We decided to participate only in Task 1 of the BioNLP shared task, mainly because of lack of time and resources.

Our event annotation method relied on various preprocessing steps and an existing state of the art dependency parser, which provided the input to the event annotator. As all the linguistic processing was performed by the preprocessor and the parser, the ideas implemented for the event annotator could remain simple while still producing reasonable results. Thus, the event annotator performed a straightforward rewriting of syntactic structures to event structures, guided by the information on the syntactic nature of events that we obtained from the training data. In this sense our system can be used as a reference for a comparison to other systems that rely completely on a dependency parser delivered analysis that is rewritten into event structures using knowledge gained from the training data.

Our system consists of a preprocessing phase that uses a pipeline of NLP tools, described in section 2 of this paper. Linguistic resources are learned automatically from the preprocessed training data (section 3). A Prolog-implemented event generator is applied directly to the preprocessing results and is guided by the relative frequencies of syntactic features provided in the resources (section 4). This is followed by a postprocessing step that removes some unlikely event structures, makes sure that all events that violate the well-formedness rules are filtered out, and finally serializes the event structures into the requested output format. In section 5 we present an illustrative example of the events generated by this approach and discuss some implications of the event model adopted in the shared task. In section 6, we describe the evaluation that we performed during the training period, the final official results on the test data, and some alternative evaluations performed in parallel to the official one. In section 7 we draw conclusions and describe future work.

2 Preprocessing

Aside from a format conversion step necessary to deal with the data provided by the shared task, the

[*]Corresponding author
[1]http://www.biocreative.org/

Proceedings of the Workshop on BioNLP: Shared Task, pages 28–36,
Boulder, Colorado, June 2009. ©2009 Association for Computational Linguistics

preprocessing phase is largely based on an existing pipeline of NLP tools, that we have developed in the OntoGene project[2] (Rinaldi et al., 2006; Rinaldi et al., 2008).

2.1 Tokenization, sentence splitting, part-of-speech tagging

For tokenization, sentence splitting, and part-of-speech (POS) tagging we used LingPipe[3]. LingPipe produces very granular tokens by default, e.g. a character sequence from abstract 10395645

> **caspase-3**-like (**CPP32/Yama/apopain**)

which contains multiple hyphens and slashes (as usual for biomedical texts) is split into 12 (rather than just 4) tokens

> **caspase, -, 3**, -, like, (, **CPP32**, /, **Yama**, /, **apopain**,)

allowing a more detailed detection of terms (shown in boldface in the examples) and trigger-words which would stay token-internal if a less granular tokenization was used.

The models used for sentence splitting and POS-tagging come with the LingPipe distribution and are trained on the GENIA corpus (Kim et al., 2003), thus providing a biomedical text aware sentence splitting and POS-tagging.

2.2 Term annotation

Correctly detecting multi-word terms in the text can substantially improve the parsing results, because long noun sequences would be grouped together and the parser can only focus on the heads of the groups and ignore the rest. In this task, however, we decided to keep things simple and rely on chunking as the only means of noun grouping.

Thus, we only annotated the terms provided by the task organizers in the *a1*-files (i.e. protein mentions). We made the assumption that terms are sequences of tokens as defined by the LingPipe tokenizer. Whereas in the vast majority of cases this coincides with the tokenization used by the organizers, there are 10 cases in the training data where this assumption is violated (e.g. 'IkappaB-alphaS32/36A' contains the term 'IkappaB-alpha' but according to LingPipe, the tokens are 'IkappaB', '-', 'alphaS32', '/', '36A').

As the last step of term annotation, we reconnected tokens which were separated by hyphens and slashes, unless the tokens were part of terms. This allowed for a more reliable processing with tools which are not optimized to deal with symbols like hyphens and slashes if these are padded with white-space.

2.3 Lemmatization using Morpha

Lemmatization was performed using Morpha (Minnen et al., 2001), which provides an accurate lemmatization given that the input contains part-of-speech information. We used the lemma information eventually only as part of the input to the dependency parser, i.e. for the other aspects of event annotation lemmas were ignored.

2.4 Chunking using LTCHUNK

Chunking can considerably reduce parsing complexity, while hardly affecting performance (Prins, 2005). In order to group contiguous sequences of nouns and verbs, we used LTCHUNK (Mikheev, 1997). LTCHUNK annotates all noun and verb groups in the sentences. A chunk is an important unit in the analysis of biomedical texts. Consider an NP chunk like

> T cell-receptor-**induced** FasL **upregulation**

which contains two event triggers, amounting to a mention of a complex event.

After applying LTCHUNK, we also detected chunk heads, with a simple algorithm — select last noun in noun groups, select last verb in verb groups. This selection is done on the basis of POS-tags.

2.5 Dependency parsing using Pro3Gres

Pro3Gres (Schneider, 2008) is a robust, deep-syntactic, broad-coverage probabilistic dependency parser, which identifies grammatical relations between the heads of chunks, including the majority of long-distance dependencies. The output is a hierarchical structure of relations (represented as the directed arrows in the example shown in figure 1).

[2] `http://www.ontogene.org/`

[3] `http://alias-i.com/lingpipe/`

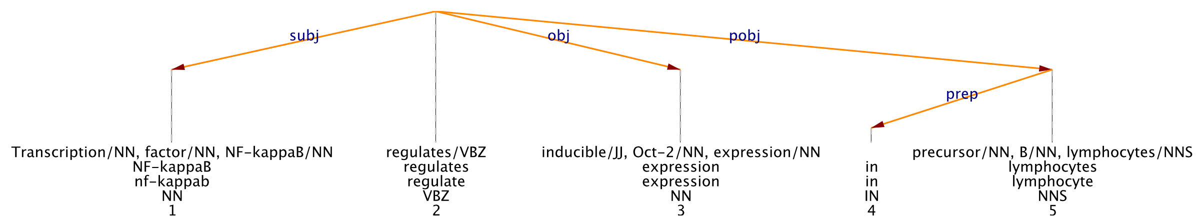

Figure 1: Dependency-syntax tree of the title of abstract 9360945: "Transcription factor NF-kappaB regulates inducible Oct-2 gene expression in precursor B lymphocytes." The dependency relations link together the heads of the 5 chunks.

The parser uses a hand-written grammar expressing linguistic competence, and a statistical language model that calculates lexicalized attachment probabilities, thus expressing linguistic performance. The parser expresses distinctions that are especially important for a predicate-argument based deep syntactic representation, as far as they are expressed in the training data generated from the Penn Treebank (Marcus et al., 1993). This includes prepositional phrase attachments, control structures, appositions, relative clause anaphora, participles, gerunds, and argument/adjunct distinctions. The dependency label set is similar to the one used in the Stanford scheme, the parser achieves state-of-the-art performance (Haverinen et al., 2008).

We have slightly adapted Pro3Gres to the biomedical domain. A class of nouns that varies considerably in the biomedical domain are relational nouns. They are syntactically marked because they can have several prepositional phrase arguments. Biomedical relational nouns like 'overexpression' or 'transcription' are absent from the Penn Treebank or rare. We have used an unsupervised approach based on (Hindle, D and Rooth, M, 1991) to learn relational nouns from Medline.

A new relation type, *hyph*, has been added to connect tokens to hyphens and slashes, and thus better deal with these characters in biomedical texts.

2.6 Preprocessor output

The preprocessor produces 5 Prolog-formatted files for each abstract. Each of these files is token-centered and affiliates a token ID with a group (either sentence, chunk, or term) that contains this token, or maps it to a syntactically related (either as the head or the dependent) token.

- **Tokens** maps each token to its lemma, POS-tag, and character offsets
- **Chunks** maps each token to its containing chunk, chunk's type (noun or verb group), and chunk's head
- **Terms** maps each token to its containing term, term's type, term's ID (assigned by the *a1*-file, or the *a2*-file in case of processing the training data)
- **Sentences** maps each sentence ID to the list of IDs of the tokens in the sentence
- **Dependencies** maps each token to its immediate head and dependent, and to the types of these dependency relations

These files are the input to the resource generator described below, and later (together with the generated resources), the input to the event annotator.

3 Resources

The 800 abstracts of the training data were used during development for the generation of three resources which are described in this section. For the official testing we used the concatenation of training and development data (i.e. 950 abstracts). The resources were generated automatically from the *a1*- and *a2*-files; and from the preprocessed version of *txt*-, *a1*- and *a2*-files. The resulting data files include frequencies of the total occurrence of an item (e.g. word, syntactic configuration) and the frequency of its occurrence in an event.

All the words in the resources were lowercased but not lemmatized. Resources were stored as Prolog-formatted files.

Frequency	Event type	Event arguments
149	Gene_expression	Theme(T)
28	Transcription	Theme(T)
2	Localization	Theme(T), AtLoc(T)
1	Positive_regulation	Theme(T)
1	Positive_regulation	Theme(E)

Table 1: Frequency distribution of the event structures that are triggered by the word form 'expressed' which in total triggered an event 181 times in the training data. 'T' means that the argument is filled by a term, 'E' means that the argument is filled by an event.

3.1 Words

The word frequencies file provides a simple probabilistic model for excluding stopwords, as we observed that many different function words sometimes triggered events in the training data. We wanted to exclude such words to obtain a better precision. The words-resource can be queried using a simple interface

```
word_to_freq(+Word, -F)
```

which maps every word to its frequency.

3.2 Event types and arguments

Using the training data, we created a mapping from each candidate trigger-word to the possible event types and the permissible event frames. A sample of this mapping is illustrated in table 1. The arguments have a type (e.g. *Theme*) but their filler is abstracted to be either 'T' (for terms) or 'E' (for events).

This resource can be queried via the interface

```
eword_to_event(+EventWord,
    -EventType, -EventArgs, -F1, -F2)
```

which maps every trigger-word to its possible event type and arguments. The returned frequencies show how often the event structure was triggered by the trigger-word, and how often the trigger-word triggered an event in total.

3.3 Domination paths between terms

The most sophisticated of the resources that we generated recorded the syntactic paths between the terms (from *a1*- and *a2*-files) observed in the training data, and counted how often these paths were present in events, connecting triggers with event argument fillers. With each term, also its type (e.g. *Positive_regulation*, *Protein*) was recorded.

For the syntactic paths, we only considered domination paths where one of the terms is the head and the other the dependent, defined as follows.

Definition 1 (Domination between chunks) *Term t_1 dominates term t_2 if $t_1 \in c_1$ and $t_2 \in c_2$ and there exists a directed syntactic path $h(c_1) \rightarrow \ldots \rightarrow h(c_2)$, where $h(\cdot)$ is the head of the given chunk.*

For example, in figure 1, the term 'regulates' dominates all the other tokens, among them the term 'expression' (which is the head of its chunk), and the *Protein*-term 'Oct-2'. Note that this definition does not require the terms to be in the chunk head position. However, this decision did not affect the results significantly.

The chunk-internal domination relation is defined for terms which are chunk-internal and thus "invisible" to the dependency parser because the parser ignores everything but the head of the chunk. This relation captures the default syntactic dependency between nouns in noun groups where the head noun usually follows its dependents.

Definition 2 (Chunk-internal domination) *Term t_1 dominates term t_2 if $t_1, t_2 \in c$ and $i(t_1) > i(t_2)$, where $i(\cdot)$ is the sequential index of the given term in the chunk.*

For example, in figure 1, in the 3rd chunk, the term 'expression' dominates the terms 'Oct-2' and 'inducible'; and furthermore, 'Oct-2' dominates 'inducible'.

The stored syntactic path is a list of dependency relations from the dependent to the head, or an empty list if both terms are in the same chunk.

Instead of domination, we also considered using the asymmetric relation of "connectedness", where two terms are connected if either of the terms dominates the other, or if both are dominated by some token in the tree. This relation, however, seemed to decrease precision much more than increase recall.

In order to query the domination resource we designed a simple query interface that allows for partially instantiated input. For example the query (where the underscores denote uninstantiated parts)

```
?- find_path_freq(bind, 'Binding',
                        _, 'Protein',
                     [modpp | _ ],
                     F1, F2).
```

asks how often there is a domination relation between the head term 'bind' if it has the type *Binding* and some dependent term with type *Protein*, such that the dependency path starts with the relation *modpp*. The frequency counts resulting from this query tell the frequency of this configuration in events (*F1*), and in total (*F2*). This information allows the computation of the conditional probability of an argument of an event given the event type, the trigger-word, the argument word, the argument type, and the syntactic path between trigger and argument.

4 Event generation

The event generation relied fully on the syntax tree and chunk information that was delivered by the preprocessing module. No fall-back to a surface co-occurrence of words was used. We only considered words and structures seen in the training data as possible parts of events. Such a design entails relatively good precision at lower recall.

For each of the generation steps described below, a probability threshold decided whether to continue the "building" of the event given the trigger-word, the event arguments template or the argument instantiation. The thresholds were set manually after some experimentation. We did not try to automatically decide the best performing thresholds. Decisions are taken locally, possibly cutting some local minima. A simple maximum-likelihood estimation (MLE) approach was used.

4.1 Trigger generation

Trigger candidates were generated from the token list of each sentence in the analyzed abstract. Figure 2 shows a browser-based visualization approach that we created as a support in our work. In the case of the training data, the annotations come the *a1*- and *a2*-files provided by the organizers. In the case of the development and test data, the annotations for the triggers are those generated by the system.

We only considered one-token trigger-words because multi-token triggers were less frequent in the training data, where only about 8% of the trigger-word forms contained a space character. Also, many of these multiword triggers contain a token that exists as a trigger on its own (e.g. 'transcriptional regulation' triggers the *Regulation*-event in the training data, as does 'regulation'), allowing us to generate a sensible event structure even if it does not match a gold standard event under the "strict equality". Tokens that had been seen to trigger an event in the training data with probability higher than 0.12 were considered further.

In MLE terms, we calculate the probability of a given token to be a trigger as follows:

$$p(Trigger \mid Token) = \frac{f(Token \land (Token = Trigger))}{f(Token)} \qquad (1)$$

4.2 Event type and arguments template generation

Next, trigger-words were mapped to event type and argument template structures. In MLE terms, we calculated the probability of an event structure (i.e. the combination of event type and arguments template) given the trigger-word.

$$p(EventStruct \mid Trigger) = \frac{f(Trigger \land EventStruct)}{f(Trigger)} \qquad (2)$$

Again, only high probability structures were considered further. We used the probability threshold of 0.25 for simple event structures (i.e. not containing nested events), and 0.1 for complex event structures (only regulation events in the shared task).

4.3 Event argument filling

The inclusion of a protein as an argument of an event was based on the syntactic domination of the trigger of the event over the term of the protein. We attempted to generate simple events of all types seen in the training data.

For complex events, the trigger-words of the main and the embedded events had to be in a domination relationship. We generated regulation-events with only 1-level embedding. Although more complex embeddings are possible (see example below), these are not very frequent.

> **prevents** T cell-receptor-**induced** FasL **upregulation**

In order to flexibly deal with sparse data, we performed a sequence of queries, one less instantiated

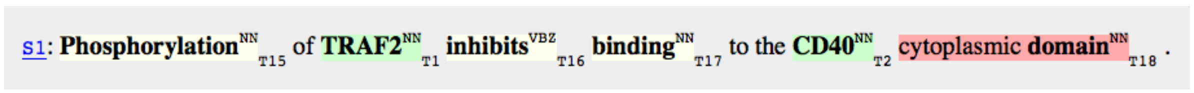
S1: Phosphorylation NN T15 of TRAF2 NN T1 inhibits VBZ T16 binding NN T17 to the CD40 NN T2 cytoplasmic domain NN T18 .

Figure 2: Example of an annotated sentence from abstract 10080948 in the training data.

than the previous one, weighted the results accordingly and calculated the weighted mean to be the final probability for including the argument.

```
find_path_freq(HWord, HType, DWord, DType, Path,
      C1_1, C2_1),
find_path_freq(_, HType, _, DType, Path,
      C1_2, C2_2),
find_path_freq(_, HType, _, DType, _,
      C1_3, C2_3)
```

In MLE terms, we calculate the probability that a syntactic configuration fills an argument slot. Syntactic configurations consist of the head word *HWord*, the head event type *HType*, the dependent word *DWord*, the dependent event type *DType*, and the syntactic path *Path* between them.

$$\begin{aligned} p(Arg \mid \mathit{HWord, HType, DWord, DType, Path}) &= \\ \frac{1}{w_1+w_2+w_3} * (\\ w_1 * \frac{f(\mathit{HWord, HType, DWord, DType, Path} \wedge \mathit{Arg})}{f(\mathit{HWord, HType, DWord, DType, Path})} &+ \\ w_2 * \frac{f(\mathit{HType, DType, Path} \wedge \mathit{Arg})}{f(\mathit{HType, DType, Path})} &+ \\ w_3 * \frac{f(\mathit{HType, DType} \wedge \mathit{Arg})}{f(\mathit{HType, DType})}) \end{aligned} \quad (3)$$

The weigths were set as $w_1 = 3$, $w_2 = 2$ and $w_3 = 1.2$. The fact that the weights decrease approximates a back-off model. Only if the final probability was higher than 0.3 the event was further considered. For complex events, we used formula 3 as given, but for simple events, where *DWord* is a protein, *DWord* was always left uninstantiated.

4.4 Postprocessing

During the postprocessing step some unlikely event structures were filtered out. This filtering is delayed until all the events have been generated, because excluding the unwanted events is difficult during creation time as sometimes extrospection is required. Also, the postprocessing step acts as a safety net that filters out well-formedness errors (e.g. argument sharing violations), thus making sure that the submission to the evaluation system is not rejected by the system. Finally, the set of generated events is serialized into the BioNLP *a2*-format.

5 Example and discussion

As an example of application of our approach, consider again the syntactic tree shown in figure 1. Our approach results in the generation of the events shown in figure 3, given that ‘regulates’, ‘inducible’, and ‘expression’ are trigger-words, and ‘Oct-2’ is an *a1*-annotated protein.

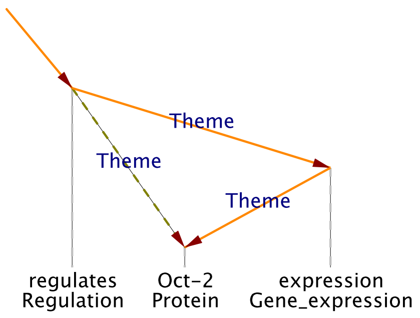

Figure 3: Visualization of two simple event structures *regulates(Oct-2)* and *expression(Oct-2)*, and a complex structure *regulates(expression(Oct-2))*.

We call events like *regulates(Oct-2)* “shortcut events”, as there exists an alternative and longer path — *regulates(expression)* and *expression(Oct-2)* — that connects the trigger to its event argument. These “shortcut events” are filtered out in the postprocessing step as unlikely events.

It is useful to observe that the particular view of event structures defined by the BioNLP shared task is by no means unchallenged. Whether nested events are necessary in a representation of biological relevant relations is a question which is open to debate. While from the linguistic perspective they do offer a more adequate representation of the content matter of the text, from the biological point of view these structures are redundant in many cases. The example used in this section is illustrative.

From the biologist’s perspective, “*A* regulates the expression of *B*” is a way to express that *A* regulates *B*. Obviously such a short-circuit is not in all cases possible, but the point is that the biologist might be interested only in the direct biological interactions, and be inclined to ignore the linguistic representation of that interaction. This is the point of view taken for example in the Protein-Protein Interaction task of the latest BioCreative competition (Krallinger et al., 2008). In that case, all linguistic structures used to better characterize the interaction are purposefully ignored, and only the bare interaction is preserved.

Since BioCreative aimed at simulating the process of database curation, and was based on datasets provided by real-word interaction databases such as IntAct (Kerrien et al., 2006) and MINT (Zanzoni et al., 2002), there is reasonable motivation for taking this alternative view into consideration. At the very least, a mapping from complex events to simple interactions should always be provided.

The difference in the approach towards interpretation of literature fragments has a direct impact on the resources used and the success of each approach. Our own development in the past couple of years has been driven by the BioCreative model (Rinaldi et al., 2008), and therefore we tended to ignore intermediate structures in protein interactions. For example, in (Schneider et al., 2009) we present a lexical resource that aims at capturing "transparent" relations, i.e. words that express a relation that from the biological point of view can be ignored because of its transitivity properties, such as "expression of Oct-2" in the example above. This resource, although certainly useful from the biological point of view, proved to be useless in the shared task, due to the different level of granularity in the representation of events.

6 Official evaluation and additional experiments

We mainly trained and evaluated using the "strict equality" evaluation criteria as our reference. The results on the development data are shown in table 2. With more relaxed equality definitions, the results were always a few percentage points better. Our results in the official testrun are shown in table 3.

Good results for some event structures (notably *Phosphorylation*) are due to the simple textual representation of these events. For example, *Phosphorylation* is always triggered by a form or derivation of 'phosphorylate', and these forms rarely trigger any other types of events. Furthermore, according to the parsed training data, the probability of a *Phosphorylation*-event, given a syntactic domination relation between a *Phosphorylation*-trigger and a protein is 0.92. Also, 56% of these domination paths are either chunk-internal or over a single *modpp* dependency relation, making them easy to detect.

In parallel to the approach used in our official submission we considered some variants, aimed at maximizing either recall or precision, as well as an alternative approach based on machine learning.

A high recall baseline method, which generates all possible event structures in a given sentence, achieves 81% recall on simple events, with precision dropping to 11%. One of the reasons why this method does not reach 100% recall is the fact that it only annotates event candidates with single-token triggers that have been seen in the training data.

The filter described in section 4.3 has a major effect on precision. If it is removed, precision drops by 11%, while the gain in recall is only 3% — recall 35.10%, precision 37.88%, F-score 36.44%. Instead, if we keep w_1 but set $w_2 = w_3 = 0$ in formula 3, precision increases to 56%, while recall drops to 27%. Increasing the probability thresholds to further improve precision results in the precision of 60% but this remains the ceiling in our experiments.

Additionally, we performed separate experiments with a machine-learning approach which considers a more varied set of features, including surface information and syntax coming from an ensemble of parsers. However, the limited time and resources available to us during the competition did not allow us to go beyond the results achieved using the approach described in detail in this paper. Since our best score on the development data was 27% (about 10% inferior to our consolidated approach), we opted for not considering this approach in our official submission.

The fact that this approach was based on a decomposition of events into their arguments led us to realize some fundamental limitations in the official evaluation measures. In particular, none of the originally implemented measures would give credit to the partial recognition of an event (i.e. correct trigger word and at least one correct argument, but not all). We contend that such partial recognition can be

Event class	Precision	Recall	F-Score	True pos.	False pos.	False neg.
Simple events	56.71	48.20	52.11	389	297	418
Complex events	38.03	19.25	25.56	189	308	793
All events	48.86	32.31	**38.90**	578	605	1211

Table 2: Results on the development data of 150 abstracts, measured using "strict equality".

Event class	gold (match)	answer (match)	Recall	Precision	F-Score
Localization	174 (31)	34 (31)	17.82	91.18	29.81
Binding	347 (102)	287 (102)	29.39	35.54	32.18
Gene_expression	722 (370)	515 (370)	51.25	71.84	59.82
Transcription	137 (28)	148 (28)	20.44	18.92	19.65
Protein_catabolism	14 (8)	16 (8)	57.14	50.00	53.33
Phosphorylation	135 (78)	84 (78)	57.78	92.86	71.23
Simple events total	1529 (617)	1084 (617)	40.35	56.92	**47.23**
Regulation	291 (29)	120 (29)	9.97	24.17	14.11
Positive_regulation	983 (138)	533 (138)	14.04	25.89	18.21
Negative_regulation	379 (55)	158 (55)	14.51	34.81	20.48
Complex events total	1653 (222)	811 (222)	13.43	27.37	**18.02**
All events total	3182 (839)	1895 (839)	26.37	44.27	**33.05**

Table 3: Results on the test data of 260 abstracts, measured using "strict equality", as reported by the BioNLP 2009 online evaluation system.

useful in a practical annotation task, and yet the official scores doubly punish such an outcome (once as a FP and once as a FN). This is a problem already observed in previous evaluation challenges, however we believe that a simple solution in this case consists in decomposing the events (for evaluation purposes) in their constituent roles and arguments. In other words, each event is given as much "weight" as its number of roles. The correct recognition of an event with two roles would therefore lead to two TP, but its partial recognition (one argument) would still lead to one TP, which we think is a more fair evaluation in case of partial recognition. Our suggestion was later implemented by the organizers as an additional scoring criteria.

7 Conclusions and future work

We have described a biological event detection method that relies on the chunk and syntactic dependency relations obtained during the preprocessing stage. No fall-back strategy that is based on e.g. surface patterns was designed for this task. This is consistent with our approach to biomedical event detection — relation extraction is entirely based on existing syntactic information about the sentences, and can be ported easily if the definition of relations and events is changed, as in the case of other competitions which use a different notion of relations (e.g. BioCreative).

As the chunker and the dependency parser form a core of the described system, their limitations and improvements have a fundamental effect on the further processing. In parallel to a thorough error analysis which can drive further development of our consolidated approach, we intend to further explore the enhanced flexibility provided by the machine learning approach briefly mentioned in section 6. In both cases, we intend to use the BioNLP shared task evaluation site as a reference in order to compare them, not only against each other, but also against the results of other participants.

Acknowledgements

This research is partially funded by the Swiss National Science Foundation (grant 100014-118396/1). Additional support is provided by Novartis Pharma AG, NITAS, Text Mining Services, CH-4002, Basel, Switzerland. The authors would like to thank the two anonymous reviewers of BioNLP 2009 for their valuable feedback.

References

[Haverinen et al.2008] Katri Haverinen, Filip Ginter, Sampo Pyysalo, and Tapio Salakoski. 2008. Accurate conversion of dependency parses: targeting the stanford scheme. In *Proceedings of Third International Symposium on Semantic Mining in Biomedicine (SMBM 2008)*, Turku, Finland.

[Hindle, D and Rooth, M1991] Hindle, D and Rooth, M. 1991. Structural Ambiguity and Lexical Relations. *Meeting of the Association for Computational Linguistics*, pages 229–236.

[Kerrien et al.2006] S. Kerrien, Y. Alam-Faruque, B. Aranda, I. Bancarz, A. Bridge, C. Derow, E. Dimmer, M. Feuermann, A. Friedrichsen, R. Huntley, C. Kohler, J. Khadake, C. Leroy, A. Liban, C. Lieftink, L. Montecchi-Palazzi, S. Orchard, J. Risse, K. Robbe, B. Roechert, D. Thorneycroft, Y. Zhang, R. Apweiler, and H. Hermjakob. 2006. IntAct — Open Source Resource for Molecular Interaction Data. *Nucleic Acids Research*.

[Kim et al.2003] J.D. Kim, T. Ohta, Y. Tateisi, and J. Tsujii. 2003. GENIA corpus — a semantically annotated corpus for bio-textmining. *Bioinformatics*, 19(1):180–182.

[Krallinger et al.2008] Martin Krallinger, Florian Leitner, Carlos Rodriguez-Penagos, and Alfonso Valencia. 2008. Overview of the protein-protein interaction annotation extraction task of BioCreative II. *Genome Biology*, 9(Suppl 2):S4.

[Marcus et al.1993] M Marcus, B Santorini, and M Marcinkiewicz. 1993. Building a Large Annotated Corpus of English: the Penn Treebank. *Computational Linguistics*, 19:313–330.

[Mikheev1997] A Mikheev. 1997. Automatic rule induction for unknown word guessing. *Computational Linguistics*, 23(3):405–423.

[Minnen et al.2001] G Minnen, J Carroll, and D Pearce. 2001. Applied morphological processing of English. *Natural Language Engineering*, 7(3):207–223.

[Prins2005] Robbert Prins. 2005. *Finite-State Pre-Processing for Natural Language Analysis*. Ph.D. thesis, Behavioral and Cognitive Neurosciences (BCN) research school, University of Groningen.

[Rinaldi et al.2006] Fabio Rinaldi, Gerold Schneider, Kaarel Kaljurand, Michael Hess, and Martin Romacker. 2006. An Environment for Relation Mining over Richly Annotated Corpora: the case of GENIA. *BMC Bioinformatics*, 7(Suppl 3):S3.

[Rinaldi et al.2008] Fabio Rinaldi, Thomas Kappeler, Kaarel Kaljurand, Gerold Schneider, Manfred Klenner, Simon Clematide, Michael Hess, Jean-Marc von Allmen, Pierre Parisot, Martin Romacker, and Therese Vachon. 2008. OntoGene in BioCreative II. *Genome Biology*, 9(Suppl 2):S13.

[Schneider et al.2009] Gerold Schneider, Kaarel Kaljurand, Thomas Kappeler, and Fabio Rinaldi. 2009. Detecting protein-protein interactions in biomedical texts using a parser and linguistic resources. In *CICLing 2009, 10th International Conference on Intelligent Text Processing and Computational Linguistics*, Mexico City, Mexico.

[Schneider2008] Gerold Schneider. 2008. *Hybrid Long-Distance Functional Dependency Parsing*. Ph.D. thesis, Faculty of Arts, University of Zurich.

[Zanzoni et al.2002] A. Zanzoni, L. Montecchi-Palazzi, M. Quondam, G. Ausiello, M. Helmer-Citterich, and G. Cesareni. 2002. MINT: a Molecular INTeraction database. *FEBS Letters*, 513(1):135–140.