Two strong baselines for the BioNLP 2009 event extraction task

Andreas Vlachos
Computer Laboratory
University of Cambridge
av308@cl.cam.ac.uk

Abstract

This paper presents two strong baselines for the BioNLP 2009 shared task on event extraction. First we re-implement a rule-based approach which allows us to explore the task and the effect of domain-adapted parsing on it. We then replace the rule-based component with support vector machine classifiers and achieve performance near the state-of-the-art without using any external resources. The good performances achieved and the relative simplicity of both approaches make them reproducible baselines. We conclude with suggestions for future work with respect to the task representation.

1 Introduction

The term *biomedical event extraction* is used to refer to tasks whose aim is the extraction of information beyond the entity level. It commonly involves recognizing actions and relations between one or more entities. The recent BioNLP 2009 shared task on event extraction (Kim et al., 2009) focused on a number of relations of varying complexity in which an event consisted of a trigger and one or more arguments. It attracted 24 submissions and provided a basis for system development. The performances ranged from 16% to 52% in F-score.

In this paper we describe two strong baseline approaches for the main task (described in Sec. 2) with a focus on annotation costs and reproducibility. Both approaches rely on a dictionary of lemmas associated with event types (Sec. 3). First we re-implement the rule-based approach of Vlachos et al. (2009) using resources provided in the shared task. While it is unlikely to reach the performance of approaches combining supervised machine learning, exploring its potential can highlight what annotated data is useful and its potential contribution to performance. Also, given its reliance on syntax, it allows us to assess the importance of syntactic parsing. Nevertheless, the performance achieved (35.39% F-score) is competitive with systems that used more annotated data and/or other resources (Sec. 5).

Building on the error analysis of the rule-based approach, we replace the rule-based component with support vector machine (SVM) classifiers trained on partial event annotation in the form of trigger-argument associations (Sec. 6). The use of a trainable classifier highlights issues concerning the suitability of the annotated data as training material. Using a simple feature representation and no external resources, the performance rises to 47.89% in F-score, which would have been second best in the shared task (Sec. 7). The error analysis suggests that future work on event extraction should look into different task representations which will allow more advanced models to demonstrate their potential (Sec. 8). Both systems shall become publically available.

2 Definition, datasets and resources

The BioNLP 2009 shared task focused on extraction of events involving proteins. Protein recognition was considered a given in order to focus the research efforts on the novel aspects of the task. Nine event types were defined in the main task, which can be broadly classified in two classes. Simple events, namely Gene_expression, Transcription, Protein_catabolism, Phosphorylation, Localization and Binding, which have proteins as their Theme argument and Regulation events, namely Positive_regulation, Negative_regulation and (unspecified) Regulation which have an obligatory Theme argument and an optional Cause argument which can be either a protein or another event. Every event has a trigger which is a contiguous textual string that can span over one or more tokens, as well as a part of a token. Triggers and arguments can be shared across events and

ID	type	trigger	Theme	Cause
E1	Neg_reg	suppressed	E2	
E2	Pos_reg	induced	E3	gp41
E3	Gene_exp	production	IL-10	

Table 1: Shared task example annotation.

the same textual string can be a trigger for events of different types. In an example demonstrating the complexity of the task: "…SQ 22536 suppressed **gp41**-induced **IL-10** production in monocytes." Participating systems, given the two proteins (in bold), need to generate the three appropriately nested events of Table 1.

While event components can reside in different sentences, we focus on events that are contained in a single sentence. Participants were not provided with resources to develop anaphora resolution components and the anaphoric phenomena involved were rather complex, as we observed in Vlachos et al. (2009). Extraction of events involving anaphoric relations inside a single sentence is still possible but it is likely to require rather complex patterns to be extracted.

The shared task involved three datasets, training, development and test, which consisted of 800, 150 and 260 abstracts respectively taken from the GENIA event corpus. Their annotation was tailored to the shared task definition. A resource made available and used by the majority of the systems was the output of four syntactic parsers:

- Bikel's (2004) re-implementation of Collins' parsing model. This parser was trained on newswire data exclusively.
- The re-ranking parser of Charniak & Johnson adapted to the biomedical domain (McClosky and Charniak, 2008). The in-domain, part-of-speech (PoS) tagger was trained on the GENIA corpus (Kim et al., 2003) and the self-training of the re-ranking module used a part of the GENIA treebank as development data.
- The C&C Combinatory Categorial Grammar (CCG) parser adapted to the biomedical domain (Rimell and Clark, 2009). The PoS tagger was trained on the GENIA corpus, while 1,000 sentences were annotated with lexical categories and added to the training data of the CCG supertagger and 600 sentences of the BioInfer corpus (Pyysalo et al., 2007) were used for parameter tuning.
- The GDep dependency parser trained for the biomedical domain in the experiments of Miyao et al. (2008). This parser was trained for the biomedical domain using the GENIA treebank.

The native Penn TreeBank output of Bikel's and McClosky's parser was converted to the Stanford Dependency (SD) collapsed dependency format (de Marneffe and Manning, 2008). The output of the CCG parser was also converted to the same dependency format, while the output of GDep was provided in a different dependency format used for the dependency parsing CoNLL 2007 shared task. From the description above, it is clear that the various parsers have different levels of adaptation to the biomedical domain. While it is difficult to assess quantitatively the actual annotation effort involved, it is possible to make some comparisons. Bikel's parser was not adapted to the domain, therefore it would be the cheapest one to deploy. McClosky and CCG used in-domain corpora annotated with PoS tags for training, while the latter using some additional annotation for lexical categories. Furthermore, they were tuned using in-domain syntactic treebanks. Therefore, they represent a more expensive option in terms of annotation cost. Finally, GDep was trained using an in-domain treebanked corpus, thus representing the alternative with the highest annotation cost.

3 Trigger extraction

We perform trigger identification using a dictionary of lemmas associated with the event type they indicate. The underlying assumption is that a particular lemma has the same semantic content in every occurrence, which results in extracting all of its occurrences as triggers of the same event type. This is clearly an over-simplification, but the restricted domain and the task definition alleviates most of the problems caused. For each lemma in the dictionary, we extract all its occurrences in the text as triggers, therefore over-generating, since not all occurrences denote a biomedical event. This can be either because they are not connected with appropriate arguments or because they are used with a sense irrelevant to the task. Both issues are being resolved at the argument identification stage since superfluous triggers should not receive arguments and not form events.

The one-sense-per-term assumption is further challenged by the fact that occurrences of the same term can denote events of different types. For example, "expression" is used as a trigger of four different event types in the training data, namely Gene_expression, Transcription, Localization and Positive_regulation. While it can be argued that in some cases this is due to annotation inconsistencies, it is generally accepted that context can alter the semantics of a token. In order to ameliorate this problem, we define the concept of *light triggers* in analogy analogy to *light verbs*. The latter are verbs whose semantics are lost when occurring in particular constructions, e.g. "make" as in "make mistakes". In the shared task, some lemmas commonly associated with a particular event type, when modified by a term associated with a different event type, denote events of the type of their modifier instead of their own. For example, "regulation" generally denotes Regulation events, unless it has a modifier of a different event type, e.g. "positive". In these cases, "regulation" becomes part of a multi-token Positive_regulation trigger (e.g. "positive regulation"). However, if the actual tokens are not adjacent, only "regulation" is annotated as a Positive_regulation trigger, which is due to the requirement that triggers are contiguous textual strings. We refer to lemmas exhibiting this behaviour as *light triggers*. Additionally, we observe that some lemmas triggered events only when modified by another lemma associated with an event type. For example, "activity" when occurring without a modifier is not considered a trigger of any event, however, when modified by "binding" then it becomes a Binding event trigger. We refer to lemmas exhibiting this behaviour as *ultra-light triggers*.[1]

In order to construct the dictionary of terms with their associated event types we use the trigger annotation from the training data, but we argue that such information could be obtained from domain experts. First, we remove the triggers encountered only once in the data in order to avoid processing non-indicative triggers. Then, we lemmatize them with *morpha* (Minnen et al., 2001). We remove prepositions and other stopwords from multi-token triggers such as "in response to" and "have a prominent increase" in order to keep only the terms denoting the event type. Then, using the single-token triggers only, we associate each lemma with its most common event type. In cases where a lemma consistently generates more that one event trigger of different types (typically one of the Simple event class and one of the Regulation class, we associate the lemma with all the relevant event types. For example, "overexpress" consistently denotes Gene_expression and Positive_regulation events. The last token of each multi-token trigger becomes a *light* trigger. Finally, if a lemma is encountered as part of a multi-token trigger of a different event type more often than with the event type associated with it as a single-token trigger, then it becomes an *ultra-light* trigger. We avoid stemming because suffixes distinguish lemmas in an important way with respect to the task. For example, "activation" denotes Positive_regulation events, while "activity" is an *ultra-light* trigger. We only keep lemmas associated at least four times with a particular event type, since below that threshold the annotation was rather inconsistent.

During testing, we attempt to match each token with one of the lemmas associated with an event type. We perform this by relaxing the matching successively, using the token lemma first and if no match is found allowing a partial match in order to deal with particles (e.g. so that "co-express" matches "express"). This process returns single-token triggers, some of which are processed further in case they are light or ultra-light using syntactic dependencies in the following stage.

4 Rule-based argument identification

In this stage, we connect the triggers extracted with appropriate arguments using rules defined with the Stanford dependency (SD) scheme (de Marneffe and Manning, 2008). We re-implement the set of rules of Vlachos et al. (2009) using the syntactic parsing resources provided by the organizers for the development data. Rule-based systems need annotated data for tuning, but unlike their supervised machine learning-based counterparts they do not learn parameters from it, thus requiring less annotated data. We consider this to be the main advantage of rule-based systems and to demonstrate this point we explicitly avoid using the training data provided. The rules define syntactic dependency paths that connect tokens containing triggers (*trigger-tokens*) with tokens containing their arguments (*arg-tokens*). For multi-token protein names, it is sufficient that a path reaches any of its tokens. For Regulation event

[1]Kilicoglu and Bergler (2009) made similar observations on the lemma "activity" without formalizing them.

class triggers we consider as *arg-tokens* not only tokens containing (parts of) protein names but also the *trigger-tokens* found in the same sentence. The rules defined are the following:

- If a *trigger-token* is the governor of an *arg-token* in subject relation (*subj*), then the latter is identified as the Theme argument of the former, e.g. "**Stat1** expresses". The only exception to this rule is that when the trigger denotes Regulation class events and the nominal subject relation (*nsubj*) is observed, the *arg-token* is identified as a Cause argument, e.g. "**gp41** induces".
- If a *trigger-token* is the governor of an *arg-token* in a prepositional relation, then the latter is identified as the Theme argument of the former, e.g. "expression of **Stat1**".
- If a *trigger-token* is the governor of an *arg-token* in modifier relation then the latter is identified as the Theme argument of the former, e.g. "**Stat1** expression". We restrict the definition of the modifier relation to subsume only the following relations: adjectival modifier (*amod*), infinitival modifier (*infmod*), participial modifier (*partmod*), adverbial modifier (*advmod*), relative clause modifier (*rcmod*), quantifier modifier (*quantmod*), temporal modifier (*tmod*) and noun compound modifier (*nn*) relations. This restriction is placed in order to avoid matches irrelevant to the task.
- If a *trigger-token* is the governor of an *arg-token* in object relation (*obj*) then the latter is identified as the Theme argument, e.g. "SQ 22536 suppressed **gp41**".
- If a Regulation event class trigger and a protein name are found in the same token, then the protein name is identified as the Cause argument, e.g. "**gp41**-induced".

A pre-processing step taken was to propagate modifier and prepositional relations over tokens that were co-ordinated or in an appositive relation. This was necessary since the SD output provided by the organizers is in the collapsed format, which treats co-ordinated tokens asymmetrically without propagating their dependencies.[2]

For each Simple or Binding trigger-argument pair, we generate a single event with the argument marked as Theme. This approach is expected to deal adequately with all event types except for Binding, which can have multiple themes. We generate Regulation events for trigger-argument pairs whose argument is a protein name or a trigger that has an already formed event. Since Regulation events can have other Regulation events as Themes or Causes, we repeat this process until no more events can be formed. Finally, at this stage we generate the required Regulation class event for triggers that consistently denote two events.

5 Rule-based system results

We report our results using the approximate span matching/approximate recursive matching variant of the evaluation. This variant allows for an event to be considered extracted correctly if its trigger is extracted with span within an one-token extension of the correct trigger span. Also in the case of nested events, events below the top-level need only their Theme argument to be correctly identified so that the top-level event is considered correct. This evaluation variant was used as the primary performance criterion in the shared task.

We first compared the performances obtained using the output of the different parsers provided by the organizers on the development data. The best F-score was achieved using McClosky (39.66%), followed by CCG (38.73%) and Bikel (36.97%). As expected, the overall performance correlates roughly with the adaptation cost involved in the development of these parsers as described in Section 2. Bikel, which is essentially unadapted, has the worst performance overall, but it would have been the cheapest to deploy. While this can be viewed as a task-based parser comparison, similar to the experiments of Miyao et al. (2008), one should be careful with the interpretation of the results. As pointed out by the authors, this type of evaluation cannot substitute a parsing evaluation against an appropriately annotated corpus since in the context of a given task only some aspects of parsing are likely to be relevant. Furthermore, in our experiments we are are not using the native output of the parsers but its conversion to the SD format. Therefore unavoidably we evaluate the conversion as well as the parsing. For this reason we avoided using the output of GDep which was not provided in this format.

Examining the lists of false positives and false negatives on the system using the McClosky

[2]The organizers re-generated the dependencies in the propagation format but we avoid using them in order to be able to compare against the shared task participants.

parser, we observe that the most common triggers of events not extracted correctly had lemmas that were included in the dictionary, such as "binding", "expression", "induction" and "activation". This suggests that most event extraction errors are due to argument identification and that using a dictionary for trigger extraction is sufficient, despite the rather strong assumptions it is based upon. Disabling the processing of light and ultra-light triggers, the performance on the development data drops to 39.28%, the main reason being the decreased recall in Binding events.

Based on the comparison performed on the development data, we run our system using the McClosky parser on the test data (Table 2). The overall performance achieved (35.39%) is relatively close to the one obtained on the development set (4% lower). This is important since rule-based systems are prone to overfitting their development data due to the way they are built. Compared to the performances achieved by the shared task participants, the system presented would be ranked seventh in overall performance. We believe this is a strong result, since it surpasses systems that used supervised machine learning methods taking advantage of the development and the training data. Restricting the comparison to rule-based systems, it would have the second best performance out of nine such systems, most of which used external knowledge sources in order to improve their performance. The best rule-based system (Kilicoglu and Bergler, 2009) had overall performance of 44.62% in F-score, ranking third overall. The main difference is that it used a much larger set of lexicalized rules (27) which were extracted using the training data. Also, heuristics were employed in order to correct syntactic parsing errors (Schuman and Bergler, 2006). While the benefits from these additional processing steps are indisputable, they involved a lot of manual work, both for rule construction as well as for the annotation of the data used to extract the rules. We argue that these performance benefits could be obtained using machine learning methods aimed at ameliorating the argument identification stage. Compared to the rule-based approach of Vlachos et al. (2009), the performance is improved substantially. The main difference between that system and the one presented here is that the former uses the domain-independent RASP parser (Briscoe et al., 2006). While its performance was reasonable (it was ranked 10th overall, 30.80% F-score), these results lag behind those reported here. Note that a direct comparison using the output of RASP is not possible since the latter uses its own syntactic dependency scheme and there is no lossless conversion to the SD scheme.

Overall, the results of this section demonstrate that the use of domain-adapted parsing is beneficial to event extraction. This is not surprising since the system presented depends heavily on the parsing output. We argue that the annotation cost of this adaptation is a good investment because, unlike the task-specific training data, improved syntactic parsing is likely to be useful for other event extraction tasks, or even other IE tasks, e.g. anaphora resolution. Therefore, we suggest that domain-adaptation of syntactic parsing should be considered first, especially in tasks that are heavily dependent on it.

6 Improving argument identification with partial annotation and support vector machines

In this section, we present an approach to argument identification which attempts to overcome the drawbacks of the rule-based approach. Following the trigger extraction stage, for each trigger combined with each of its candidate arguments we create a classification instance. The classification task is to assign the correct argument type to the instance. Therefore, we construct a binary classifier which determines whether a protein name is the Theme argument of a Simple or a Binding trigger (*ThemePositive* or *ThemeNegative*) and a ternary classifier which determines whether a protein name or another trigger (and as consequence its associated events) is the Theme or the Cause argument of a Regulation trigger (*RegThemePositive*, *RegCausePositive*, *RegNegative*).

In order to acquire labeled instances for training, we decompose the gold standard (GS) events into multiple events with single arguments. In cases of events being arguments to Regulation events, the former are replaced by their triggers. We match the triggers extracted with those included in the gold standard, ignoring the event type annotation. Since we identify single-token triggers, we replicate the approximate span matching used in evaluation in order to achieve better coverage. If the instance being considered has a Simple or a Binding trigger, and if the pair is in-

	Rules (MC)			SVM (MC+CCG)		
Event Type/Class	recall	precision	F-score	recall	precision	F-score
Gene_expression	46.54	78.50	58.43	61.63	82.26	70.47
Transcription	26.28	28.57	27.38	29.93	62.12	40.39
Protein_catabolism	28.57	100.00	44.44	42.86	85.71	57.14
Phosphorylation	65.19	82.24	72.73	78.52	91.38	84.46
Localization	32.18	88.89	47.26	40.80	95.95	57.26
Simple (total)	43.99	71.43	54.45	56.60	83.21	67.37
Binding	20.46	38.17	26.64	29.11	45.29	35.44
Regulation	15.81	23.47	18.89	23.71	39.20	29.55
Positive	21.16	33.02	25.79	37.03	43.65	40.07
Negative	17.15	29.41	21.67	30.34	40.35	34.64
Regulation (total)	19.30	30.47	23.63	33.15	42.32	37.18
Total	28.60	46.40	35.39	41.42	56.76	47.89

Table 2: Performance of the rule-based and the SVM-based systems on the test data. Each horizontal corresponds to an event type or class. Binding events are not included in the Simple class aggregate performance because they can have multiple Themes.

cluded in the GS then it is labeled as *ThemePositive*, else it labeled as *ThemeNegative*. If the instance being considered has a Regulation trigger that has been matched with a GS trigger, and if its argument is a protein name and their pair is included in the GS then it is labeled according to the latter (*Reg{Theme/Cause}Positive*), else, if not found in the GS it is labeled as *RegThemeNegative*. The same process is followed if the argument is an event trigger which has been matched with a GS trigger. We consider only Regulation triggers that are matched in the GS in order to avoid valid *RegCausePositive* instances being labeled as *RegNegative*. Recall that the Cause argument is optional, while the Theme is obligatory for Regulation events. This means that if an appropriate Theme argument is not present, then it is possible that a Cause argument that is present is not annotated. Similarly, when considering event triggers as arguments, we acquire labels only for instances involving triggers that were annotated in the GS. Since triggers without an appropriate Theme are not annotated in the gold standard, it is possible that a valid *RegThemePositive* or *RegCausePositive* is labeled as *RegNegative* instance not because of the actual relation between the trigger and the argument but because the argument did not have an appropriate Theme present. In the example mentioned in Sec. 2, if "IL-10" was replaced by "protein" then none of the events would be annotated. We argue that a human annotator would produce these annotations implicitly, and that this partial (with respect to the task definition) annotation scheme allows the encoding of this information in a more flexible way. Also, this is likely to be a more efficient way to use the annotation time, since annotators would be requested to annotate pre-determined trigger-argument pairs instead of searching for events from scratch, given only the protein name annotation.

For training data generation we consider the triggers extracted using the dictionary instead of those in the GS. This process is certain to introduce some noise as some triggers might be omitted due to limited dictionary coverage. If the event type determined by the dictionary is incorrect, this is unlikely to affect the argument identification process, since the latter is dependent on the lemma of the trigger rather than its type. For example, the Theme argument of the trigger "expression" is unlikely to depend on whether the event denoted is Gene_expression or Transcription.

The labeled instance acquisition process described results in 9,699 binary and 10,541 ternary labels compared to 6,607 triggers and 9,597 events annotated in the training data provided. However, it must be pointed out that in the shared task annotation scheme negative instances are annotated implicitly, i.e. non-events are not annotated. If we consider only the positive instances, then the annotation scheme describeed results in 3,517 *ThemePositive* and 3,933 *Reg{Theme/Cause}Positive* instances, which are simpler since they do not need require textual span and event type specification.

For feature extraction, we find the shortest dependency path connecting each trigger-argument pair using Dijkstra's algorithm. We allow paths to follow either dependency direction by incorporating the direction in the dependency labels. Apart from the dependency path, we extract as features the trigger-token, the trigger event type and the argument type (event type if the argument is a trigger or *Entity* in case of protein names). We filtered the training set considering only instances in which the trigger was at a maximum distance of 4 dependencies away from the argument, since longer paths were too sparse to be useful in classifying unseen instances. At classification time, we consider as {*Theme/Reg*}*Negative* any instances in which the dependency path has not been encountered in the training data, as well as instances without a dependency path connecting trigger and argument. This is necessary in order to avoid instances being classified only on the basis of the trigger-token and the argument type. After the classifier has assigned labels to the trigger-argument pairs, we construct events as described in Sec. 4. In cases where it is unclear (to the classifier) which is the trigger and which is the argument in a given pair of Regulation event triggers the process can result in cyclic dependencies. We resolve them using the confidence of the classifier for each decision by removing the least confident *RegThemePositive* or *RegCausePositive* assignment.

7 SVM-based system results

In our experiments we used the LIBSVM toolkit (Chang and Lin, 2001) which provides an implementation of Support Vector Machines with various kernels and uses the one-against-one scheme for multiclass problems. In all experiments, the Gaussian kernel was used in order to capture potential non-linear feature combinations, e.g. cases where the combination of dependency path and trigger-token would result in a different decision rather than each of them independently. Preliminary experiments with the linear kernel confirmed this expectation.

We focused on using the output of the two domain-adapted parsers, namely CCG and McClosky. The reason for this is that, as argued in Sec. 5, given the importance of syntactic parsing to event extraction one should consider domain adaptation of syntactic parsing before developing task-specific training resources. We first compared the performances obtained using the output of the different parsers provided by the organizers using the development data. The main observation is that, using either parser, the results are much improved compared to those reported in Sec. 5, by approximately eight percentage points in F-score in either case (46.49% and 47.40% F-score for CCG and McClosky respectively). Most of the improvement is due to higher recall, suggesting that the argument identification component is able to learn patterns that are relevant to the task. Overall, using the output of CCG results in higher precision, while McClosky results in higher recall. These parsers have different theoretical foundations, therefore they are expected to make different errors. In an effort to take advantage of both parsers simultaneously, we combined them by adding for each trigger-argument pair the dependency paths extracted by both parsers. This improved performance further to 49.35% F-score.

We then run the system combining the two parsers on the test data, obtaining the results presented in Table 2. Overall, the system presented would have the second best performance in the shared task achieving 41.42%/56.76%/47.89% in Recall/Precision/F-score. The top system (Bjorne et al., 2009) achieved 46.73%/58.48%/51.95% (R/P/F). It followed a machine learning approach to trigger extraction which, while it is likely to be responsible for the performance difference observed when compared to the other participating systems, requires explicit trigger annotation, thus being more expensive. Furthermore, we argue that the data provided by the organizers are not suitable to train a trigger extractor, since only triggers participating in events are annotated, and semantically valid triggers without appropriate arguments present are ignored. We hypothesize that this is the reason the authors had to adjust the decisions of their SVM classifiers.

The second best system (Buyko et al., 2009) achieved 45.82%/47.52%/46.66% (R/P/F) using many external knowledge sources such as the Gene Ontology Annotation database, the Universal Protein Resourceand the Medical Subject Headings thesaurus. While the use of these resources and their successful usage is commendable, we believe it is important that the system presented achieves comparable performance using fewer resources.

Furthermore, joint inference models such as

Markov Logic Networks were applied to the BioNLP 2009 event extraction shared task by Riedel et al. (2009) and were ranked fourth. This result was improved upon recently by Poon and Vanderwende (2010) who achieved 50% F-score, 2.11 percentage points better than the result achieved in this work. Such models have great potential for event extraction and we believe that they can benefit from the insights presented here. Finally, despite the fact that we used the same experimental setup as the shared task participants, we do not consider our results are directly comparable to theirs since we did not work under the same time constraints and we profited from their experiences.

8 Discussion

Our error analysis on the output of the best system on the development data discouraged us from pursuing further improvements. Echoing the observations of Buyko et al. (2009), we found that annotation inconsistency was affecting our results significantly. In many cases the event triggers annotated in the development data were rather misleading, e.g. "negative" as a Gene_expression event trigger (abstract 8622883), "increase the stability" as a Positive_regulation event trigger (abstract 8626752), "disappearance" as a Binding event trigger (abstract 10455128). Finally, some events were ignored by the annotation, such as "regulation of **thymidine kinase**" (abstract 8622883).

An additional complication is that events that are annotated due to anaphoric linking can have a disproportionate effect on the scores. In an example from abstract 9794375: "CD3, **CD2**, and **CD28** are functionally distinct receptors on T lymphocytes. Engagement of any of these receptors induces the rapid tyrosine phosphorylation of a shared group of intracellular signaling proteins, including **Vav**, **Cbl**, p85 phosphoinositide 3-kinase, and the **Src** family kinases **Lck** and **Fyn**." Failing to recognize the anaphoric Binding events involving proteins "CD2" and "CD28", an otherwise perfect system would receive two false negatives for the Binding events, eight false negatives for the missing Positive_regulation events due to the missing Causes and four false positives for the incomplete Positive_regulation events extracted.

Despite this criticism, we believe that the BioNLP 2009 shared task on event extraction was a big step forward for biomedical information extraction and we are grateful to the organizers for the effort and resources provided, without which the research presented here would not have been possible. The performances achieved in the main Task1 ranged from 16% to 52% in F-score, suggesting improvements in task definition, data annotation and participating systems compared to previous community-wide efforts. Indicatively, in the protein-protein interaction pair subtask of BioCreative II (Krallinger et al., 2008) the annotated datasets provided were produced by extracting curation information from relevant databases. This meant that there was no text-bound annotation, thus making the application and evaluation of existing NLP techniques difficult, resulting in rather low performances. The best performance achieved was 29% in F-score, while many of the teams scored below 10%.

However, we believe that future work should look at improving the annotation in order to be able to assess the progress in the systems developed. In particular, we argue that we should move towards a dependency-based representation, similar to the one introduced by Surdeanu et al. (2008) for joint syntactic parsing and semantic role labeling. Such representation can express the nested nature of the events and evaluate the dependencies between them directly. Furthermore, given the importance of syntactic parsing via syntactic dependencies to event extraction, it would be interesting to see how performing these tasks jointly would help improve the performance. A dependency-based representation would also allow for non-contiguous event components, as well as more complex phenomena such as the *light triggers* discussed earlier.

9 Conclusions

In this paper we focused on the BioNLP 2009 shared task on event extraction. We developed two systems, a rule-based one that does not require training data and a SVM-based one which achieves near state-of-the-art performance. The good performances achieved and their reliance on shared task resources exclusively makes them reproducible and strong baselines for future work. Furthermore, we demonstrated the importance of domain adaptation of syntactic parsing for event extraction. Finally, based on our error analysis we suggest future directions for event extraction with respect to the task representation.

Acknowledgements

The author would like to thank Ted Briscoe, Mark Craven and the three anonymous reviewers for their feedback.

References

Daniel M. Bikel. 2004. Intricacies of Collins' parsing model. *Computational Linguistics*, 30(4):479–511.

Jari Bjorne, Juho Heimonen, Filip Ginter, Antti Airola, Tapio Pahikkala, and Tapio Salakoski. 2009. Extracting complex biological events with rich graph-based feature sets. In *Proceedings of the BioNLP'09 Shared Task on Event Extraction*, pages 10–18.

Ted Briscoe, John Carroll, and Rebecca Watson. 2006. The second release of the RASP system. In *Proceedings of the COLING/ACL Interactive presentation sessions*, pages 77–80, Morristown, NJ, USA. Association for Computational Linguistics.

Ekaterina Buyko, Erik Faessler, Joachim Wermter, and Udo Hahn. 2009. Event extraction from trimmed dependency graphs. In *Proceedings of the BioNLP 2009 Workshop Companion Volume for Shared Task*, pages 19–27.

Chih-Chung Chang and Chih-Jen Lin, 2001. *LIBSVM: a library for support vector machines*. Software available at http://www.csie.ntu.edu.tw/˜cjlin/libsvm.

Marie-Catherine de Marneffe and Christopher D. Manning. 2008. The stanford typed dependencies representation. In *CrossParser '08: Coling 2008: Proceedings of the Workshop on Cross-Framework and Cross-Domain Parser Evaluation*, pages 1–8.

Halil Kilicoglu and Sabine Bergler. 2009. Syntactic dependency based heuristics for biological event extraction. In *Proceedings of the BioNLP 2009 Workshop Companion Volume for Shared Task*, pages 119–127.

Jin-Dong Kim, Tomoko Ohta, Yuka Tateisi, and Jun'ichi Tsujii. 2003. GENIA corpus - a semantically annotated corpus for bio-textmining. In *Bioinformatics*, volume 19, Suppl. 1, pages 180–182.

Jin-Dong Kim, Tomoko Ohta, Sampo Pyysalo, Yoshinobu Kano, and Jun'ichi Tsujii. 2009. Overview of BioNLP'09 shared task on event extraction. In *BioNLP '09: Proceedings of the Workshop on BioNLP*, pages 1–9.

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.

David McClosky and Eugene Charniak. 2008. Self-training for biomedical parsing. In *Proceedings of the 46th Annual Meeting of the Association of Computational Linguistics: Human Language Technologies*, pages 101–104.

Guido Minnen, John Carroll, and Darren Pearce. 2001. Applied morphological processing of English. *Natural Language Engineering*, 7(3):207–223.

Yusuke Miyao, Rune Sætre, Kenji Sagae, Takuya Matsuzaki, and Jun'ichi Tsujii. 2008. Task-oriented evaluation of syntactic parsers and their representations. In *Proceedings of the 46th Annual Meeting of the Association of Computational Linguistics: Human Language Technologies*, pages 46–54.

Hoifung Poon and Lucy Vanderwende. 2010. Joint inference for knowledge extraction from biomedical literature. In *Proceedings of the North American Chapter of the Association for Computational Linguistics - Human Language Technologies 2010 conference*.

Sampo Pyysalo, Filip Ginter, Juho Heimonen, Jari Bjorne, Jorma Boberg, Jouni Jarvinen, and Tapio Salakoski. 2007. BioInfer: a corpus for information extraction in the biomedical domain. *BMC Bioinformatics*, 8(1):50+.

Sebastian Riedel, Hong-Woo Chun, Toshihisa Takagi, and Jun'ichi Tsujii. 2009. A Markov Logic Approach to Bio-Molecular Event Extraction. In *Proceedings of the BioNLP 2009 Workshop Companion Volume for Shared Task*, pages 41–49.

Laura Rimell and Stephen Clark. 2009. Porting a lexicalized-grammar parser to the biomedical domain. *Journal of Biomedical Informatics*, 42(5):852–865.

Jonathan Schuman and Sabine Bergler. 2006. Post-nominal prepositional phrase attachment in proteomics. In *Proceedings of the Workshop on Linking Natural Language Processing and Biology*, pages 82–89.

Mihai Surdeanu, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. 2008. The CoNLL-2008 shared task on joint parsing of syntactic and semantic dependencies. In *CoNLL 2008: Proceedings of the Twelfth Conference on Natural Language Learning*, pages 159–177.

Andreas Vlachos, Paula Buttery, Diarmuid Ó Séaghdha, and Ted Briscoe. 2009. Biomedical event extraction without training data. In *Proceedings of the BioNLP 2009 Workshop Companion Volume for Shared Task*, pages 37–40.