NAACL HLT 2009

BioNLP 2009

Companion Volume: Shared Task on Event Extraction

June 5, 2009
Boulder, Colorado

Production and Manufacturing by
Omnipress Inc.
2600 Anderson Street
Madison, WI 53707
USA

Order copies of this and other ACL proceedings from:

Association for Computational Linguistics (ACL)
209 N. Eighth Street
Stroudsburg, PA 18360
USA
Tel: +1-570-476-8006
Fax: +1-570-476-0860
acl@aclweb.org

ISBN 978-1-932432-44-2

Introduction

The need for automatic processing of the rapidly increasing body of publications describing results in molecular biology continues to drive efforts in Biomedical natural language processing (BioNLP). Until recently, domain efforts have largely concentrated on foundational tasks such as entity recognition and relatively simple information extraction targets such as interacting entity pairs. By contrast, biological research increasingly aims to create detailed descriptions of complex processes in biological systems. To respond to the needs of such research, it is necessary to develop BioNLP methods that are able to process more fine-grained representations. The BioNLP'09 Shared Task is the first community-wide step in this direction.

Shared tasks have a strong tradition in the BioNLP community. The TREC Genomics, KDD cup, JNLBPA, LLL and BioCreative tasks have focused the efforts of the community on timely challenges in the domain, both establishing the capabilities and problem points of current systems as well as advancing the state of the art in various areas of biomedical text mining. These are also the goals of the present shared task. The focus on a rich representation of extracted information in the BioNLP'09 Shared Task can also be seen as natural continuation of the succession of previous tasks.

The BioNLP'09 Shared Task targets event extraction following a model similar to those currently applied in the wider IE community. Corpus resources supporting this type of representation have only recently become available in the domain, and the task thus represents a novel challenge to much of the community. Despite its novelty and relatively complex task settings, the BioNLP'09 Shared Task met with an enthusiastic response from the community: more than 40 teams registered their preliminary interest, and 24 teams submitted final results. Thanks to the efforts of the participants and the shared task program committee in reviewing, we have the pleasure of presenting these proceedings of 19 manuscripts accepted for presentation at the shared task session of the workshop.

Table of Contents

Conference Program

Friday, June 5, 2009 (continued)

12:30–14:00 Lunch

Session 3: Poster presentations

14:00–14:15 Introduction to poster presentations

14:15–15:30 Poster presentations

Molecular event extraction from Link Grammar parse trees
Jörg Hakenberg, Illes Solt, Domonkos Tikk, Luis Tari, Astrid Rheinländer, Nguyen Quang Long, Graciela Gonzalez and Ulf Leser

Tunable Domain-Independent Event Extraction in the MIRA Framework
Georgi Georgiev, Kuzman Ganchev, Vassil Momchev, Deyan Peychev, Preslav Nakov and Angus Roberts

BioEve: Bio-Molecular Event Extraction from Text Using Semantic Classification and Dependency Parsing
Syed Toufeeq Ahmed, Radhika Nair, Chintan Patel and Hasan Davulcu

From Protein-Protein Interaction to Molecular Event Extraction
Rune Sætre, Makoto Miwa, Kazuhiro Yoshida and Jun'ichi Tsujii

A Multi-Phase Approach to Biomedical Event Extraction
Hyoung-Gyu Lee, Han-Cheol Cho, Min-Jeong Kim, Joo-Young Lee, Gumwon Hong and Hae-Chang Rim

Supervised Classification for Extracting Biomedical Events
Arzucan Ozgur and Dragomir Radev

Biomedical Event Detection using Rules, Conditional Random Fields and Parse Tree Distances
Farzaneh Sarafraz, James Eales, Reza Mohammadi, Jonathan Dickerson, David Robertson and Goran Nenadic

15:30–16:00 Coffee break

Overview of BioNLP'09 Shared Task on Event Extraction

Jin-Dong Kim[*] **Tomoko Ohta**[*] **Sampo Pyysalo**[*] **Yoshinobu Kano**[*] **Jun'ichi Tsujii**[*†‡]
[*]Department of Computer Science, University of Tokyo, Tokyo, Japan
[†]School of Computer Science, University of Manchester, Manchester, UK
[‡]National Centre for Text Mining, University of Manchester, Manchester, UK
{jdkim,okap,smp,kano,tsujii}@is.s.u-tokyo.ac.jp

Abstract

The paper presents the design and implementation of the BioNLP'09 Shared Task, and reports the final results with analysis. The shared task consists of three sub-tasks, each of which addresses bio-molecular event extraction at a different level of specificity. The data was developed based on the GENIA event corpus. The shared task was run over 12 weeks, drawing initial interest from 42 teams. Of these teams, 24 submitted final results. The evaluation results are encouraging, indicating that state-of-the-art performance is approaching a practically applicable level and revealing some remaining challenges.

1 Introduction

The history of text mining (*TM*) shows that shared tasks based on carefully curated resources, such as those organized in the MUC (Chinchor, 1998), TREC (Voorhees, 2007) and ACE (Strassel et al., 2008) events, have significantly contributed to the progress of their respective fields. This has also been the case in *bio-TM*. Examples include the TREC Genomics track (Hersh et al., 2007), JNLPBA (Kim et al., 2004), LLL (Nédellec, 2005), and BioCreative (Hirschman et al., 2007). While the first two addressed *bio-IR* (information retrieval) and *bio-NER* (named entity recognition), respectively, the last two focused on *bio-IE* (information extraction), seeking relations between bio-molecules. With the emergence of NER systems with performance capable of supporting practical applications, the recent interest of the bio-TM community is shifting toward IE.

Similarly to LLL and BioCreative, the BioNLP'09 Shared Task (the BioNLP task, hereafter) also addresses bio-IE, but takes a definitive step further toward finer-grained IE. While LLL and BioCreative focus on a rather simple representation of relations of bio-molecules, i.e. protein-protein interactions (PPI), the BioNLP task concerns the detailed behavior of bio-molecules, characterized as bio-molecular events (*bio-events*). The difference in focus is motivated in part by different applications envisioned as being supported by the IE methods. For example, BioCreative aims to support curation of PPI databases such as MINT (Chatr-aryamontri et al., 2007), for a long time one of the primary tasks of bioinformatics. The BioNLP task aims to support the development of more detailed and structured databases, e.g. pathway (Bader et al., 2006) or Gene Ontology Annotation (GOA) (Camon et al., 2004) databases, which are gaining increasing interest in bioinformatics research in response to recent advances in molecular biology.

As the first shared task of its type, the BioNLP task aimed to define a bounded, well-defined bio-event extraction task, considering both the actual needs and the state of the art in *bio-TM* technology and to pursue it as a community-wide effort. The key challenge was in finding a good balance between the utility and the feasibility of the task, which was also limited by the resources available. Special consideration was given to providing evaluation at diverse levels and aspects, so that the results can drive continuous efforts in relevant directions. The paper discusses the design and implementation of the BioNLP task, and reports the results with analysis.

Proceedings of the Workshop on BioNLP: Shared Task, pages 1–9,
Boulder, Colorado, June 2009. ©2009 Association for Computational Linguistics

Type	Primary Args.	Second. Args.
Gene_expression	T(P)	
Transcription	T(P)	
Protein_catabolism	T(P)	
Phosphorylation	T(P)	Site
Localization	T(P)	AtLoc, ToLoc
Binding	T(P)+	Site+
Regulation	T(P/Ev), C(P/Ev)	Site, CSite
Positive_regulation	T(P/Ev), C(P/Ev)	Site, CSite
Negative_regulation	T(P/Ev), C(P/Ev)	Site, CSite

Table 1: Event types and their arguments. The type of the filler entity is specified in parenthesis. The filler entity of the secondary arguments are all of *Entity* type which represents any entity but proteins: T=Theme, C=Cause, P=Protein, Ev=Event.

2 Task setting

To focus efforts on the novel aspects of the event extraction task, is was assumed that named entity recognition has already been performed and the task was begun with a given set of gold protein annotation. This is the only feature of the task setting that notably detracts from its realism. However, given that state-of-the-art protein annotation methods show a practically applicable level of performance, i.e. 88% F-score (Wilbur et al., 2007), we believe the choice is reasonable and has several advantages, including focus on event extraction and effective evaluation and analysis.

2.1 Target event types

Table 1 shows the event types addressed in the BioNLP task. The event types were selected from the GENIA ontology, with consideration given to their importance and the number of annotated instances in the GENIA corpus. The selected event types all concern protein biology, implying that they take proteins as their theme. The first three types concern protein metabolism, i.e. protein production and breakdown. Phosphorylation is a representative protein modification event, and Localization and Binding are representative fundamental molecular events. Regulation (including its sub-types, Positive and Negative_regulation) represents regulatory events and causal relations. The last five are universal but frequently occur on proteins. For the biological interpretation of the event types, readers are referred to Gene Ontology (GO) and the GENIA ontology.

The failure of p65 translocation to the nucleus . . .

T3	(Protein, 40-46)
T2	**(Localization, 19-32)**
E1	**(Type:T2, Theme:T3**, ToLoc:T1**)**
T1	(Entity, 15-18)
M1	(Negation E1)

Figure 1: Example event annotation. The protein annotation T3 is given as a starting point. The extraction of annotation in bold is required for Task 1, T1 and the ToLoc:T1 argument for Task 2, and M1 for Task 3.

As shown in Table 1, the theme or themes of all events are considered primary arguments, that is, arguments that are critical to identifying the event. For regulation events, the entity or event stated as the *cause* of the regulation is also regarded as a primary argument. For some event types, other arguments detailing of the events are also defined (*Secondary Args.* in Table 1).

From a computational point of view, the event types represent different levels of complexity. When only primary arguments are considered, the first five event types require only unary arguments, and the task can be cast as relation extraction between a predicate (event trigger) and an argument (Protein). The Binding type is more complex in requiring the detection of an arbitrary number of arguments. Regulation events always take a Theme argument and, when expressed, also a Cause argument. Note that a Regulation event may take another event as its theme or cause, a unique feature of the BioNLP task compared to other event extraction tasks, e.g. ACE.

2.2 Representation

In the BioNLP task, events are expressed using three different types of entities. *Text-bound entities* (*t-entities* hereafter) are represented as text spans with associated class information. The t-entities include event triggers (*Localization*, *Binding*, etc), protein references (*Protein*) and references to other entities (*Entity*). A t-entity is represented by a pair, (*entity-type*, *text-span*), and assigned an id with the prefix "T", e.g. T1–T3 in Figure 1. An *event* is expressed as an n-tuple of typed t-entities, and has a id with prefix "E", e.g. E1. An *event modification* is expressed by a pair, (*predicate-negation-or-speculation*, *event-id*), and has an id with prefix "M", e.g. M1.

Item	Training	Devel.	Test
Abstract	800	150	260
Sentence	7,449	1,450	2,447
Word	176,146	33,937	57,367
Event	8,597 / 8,615	1,809 / 1,815	3,182 / 3,193

Table 2: Statistics of the data sets. For events, Task1/Task2 shown separately as secondary arguments may introduce additional differentiation of events.

2.3 Subtasks

The BioNLP task targets semantically rich event extraction, involving the extraction of several different classes of information. To facilitate evaluation on different aspects of the overall task, the task is divided to three sub-tasks addressing event extraction at different levels of specificity.

Task 1. Core event detection detection of typed, text-bound events and assignment of given proteins as their primary arguments.

Task 2. Event enrichment recognition of secondary arguments that further specify the events extracted in Task 1.

Task 3. Negation/Speculation detection detection of negations and speculation statements concerning extracted events.

Task 1 serves as the backbone of the shared task and is mandatory for all participants. Task 2 involves the recognition of *Entity* type t-entities and assignment of those as secondary event arguments. Task 3 addresses the recognition of negated or speculatively expressed events without specific binding to text. An example is given in Fig. 1.

3 Data preparation

The BioNLP task data were prepared based on the GENIA event corpus. The data for the training and development sets were derived from the publicly available event corpus (Kim et al., 2008), and the data for the test set from an unpublished portion of the corpus. Table 2 shows statistics of the data sets.

For data preparation, in addition to filtering out irrelevant annotations from the original GENIA corpus, some new types of annotation were added to make the event annotation more appropriate for the purposes of the shared task. The following sections describe the key changes to the corpus.

3.1 Gene-or-gene-product annotation

The named entity (NE) annotation of the GENIA corpus has been somewhat controversial due to differences in annotation principles compared to other biomedical NE corpora. For instance, the NE annotation in the widely applied GENETAG corpus (Tanabe et al., 2005) does not differentiate proteins from genes, while GENIA annotation does. Such differences have caused significant inconsistency in methods and resources following different annotation schemes. To remove or reduce the inconsistency, GENETAG-style NE annotation, which we term gene-or-gene-product (GGP) annotation, has been added to the GENIA corpus, with appropriate revision of the original annotation. For details, we refer to (Ohta et al., 2009). The NE annotation used in the BioNLP task data is based on this annotation.

3.2 Argument revision

The GENIA event annotation was made based on the GENIA event ontology, which uses a loose typing system for the arguments of each event class. For example, in Figure 2(a), it is expressed that the binding event involves two proteins, TRAF2 and CD40, and that, in the case of CD40, its cytoplasmic domain takes part in the binding. Without constraints on the type of theme arguments, the following two annotations are both legitimate:

(Type:Binding, Theme:*TRAF2*, Theme:*CD40*)
(Type:Binding, Theme:*TRAF2*,
Theme:*CD40 cytoplasmic domain*)

The two can be seen as specifying the same event at different levels of specificity[1]. Although both alternatives are reasonable, the need to have consistent training and evaluation data requires a consistent choice to be made for the shared task.

Thus, we fix the types of all non-event primary arguments to be proteins (specifically GGPs). For GENIA event annotations involving themes other than proteins, additional argument types were introduced, for example, as follows:

[1]In the GENIA event annotation guidelines, annotators are instructed to choose the more specific alternative, thus the second alternative for the example case in Fig. 2(a).

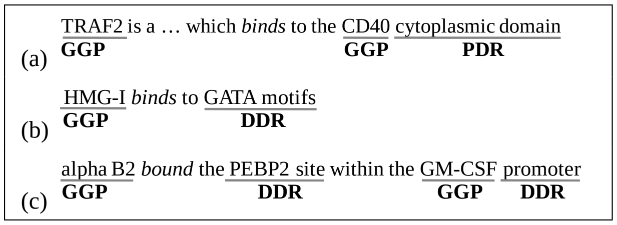

(a)
TRAF2 is a … which binds to the CD40 cytoplasmic domain
GGP GGP PDR
(b)
HMG-I binds to GATA motifs
GGP DDR
(c)
alpha B2 bound the PEBP2 site within the GM-CSF promoter
GGP DDR GGP DDR

Figure 2: Entity annotation to example sentences from (a) PMID10080948, (b) PMID7575565, and (c) PMID7605990 (simplified).

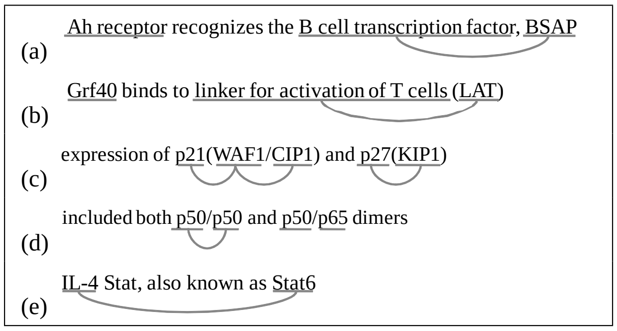

(a)
Ah receptor recognizes the B cell transcription factor, BSAP
(b)
Grf40 binds to linker for activation of T cells (LAT)
(c)
expression of p21(WAF1/CIP1) and p27(KIP1)
(d)
included both p50/p50 and p50/p65 dimers
(e)
IL-4 Stat, also known as Stat6

Figure 3: Equivalent entities in example sentences from (a) PMID7541987 (simplified), (b) PMID10224278, (c) PMID10090931, (d) PMID9243743, (e) PMID7635985.

(Type:Binding, Theme1:*TRAF2*, Theme2:*CD40*,
Site2:*cytoplasmic domain*)

Note that the protein, CD40, and its domain, cytoplasmic domain, are associated by argument numbering. To resolve issues related to the mapping between proteins and related entities systematically, we introduced partial static relation annotation for relations such as Part-Whole, drawing in part on similar annotation of the BioInfer corpus (Pyysalo et al., 2007). For details of this part of the revision process, we refer to (Pyysalo et al., 2009).

Figure 2 shows some challenging cases. In (b), the site *GATA motifs* is not identified as an argument of the binding event, because the protein containing it is not stated. In (c), among the two sites (*PEBP2 site* and *promoter*) of the gene *GM-CSF*, only the more specific one, *PEBP2*, is annotated.

3.3 Equivalent entity references

Alternative names for the same object are frequently introduced in biomedical texts, typically through apposition. This is illustrated in Figure 3(a), where the two expressions *B cell transcription factor* and *BSAP* are in apposition and refer to the same protein. Consequently, in this case the following two annotations represent the same event:

(Type:Binding, Theme:*Ah receptor*,
Theme:*B cell transcription factor*)
(Type:Binding, Theme:*Ah receptor*, Theme:*BSAP*)

In the GENIA event corpus only one of these is annotated, with preference given to shorter names over longer descriptive ones. Thus of the above example events, the latter would be annotated. However, as both express the same event, in the shared task evaluation either alternative was accepted as correct extraction of the event. In order to implement this aspect of the evaluation, expressions of equivalent entities were annotated as follows:

Eq (*B cell transcription factor*, *BSAP*)

The equivalent entity annotation in the revised GENIA corpus covers also cases other than simple apposition, illustrated in Figure 3. A frequent case in biomedical literature involves use of the slash symbol ("/") to state synonyms. The slash symbol is ambiguous as it is used also to indicate dimerized proteins. In the case of *p50/p50*, the two *p50* are annotated as equivalent because they represent the same proteins at the same state. Note that although rare, also explicitly introduced aliases are annotated, as in Figure 3(e).

4 Evaluation

For the evaluation, the participants were given the test data with gold annotation only for proteins. The evaluation was then carried out by comparing the annotation predicted by each participant to the gold annotation. For the comparison, equality of annotations is defined as described in Section 4.1. The evaluation results are reported using the standard recall/precision/f-score metrics, under different criteria defined through the equalities.

4.1 Equalities and Strict matching

Equality of events is defined as follows:

Event Equality equality holds between any two events when (1) the event types are the same, (2) the event triggers are the same, and (3) the arguments are fully matched.

A full matching of arguments between two events means there is a perfect 1-to-1 mapping between the two sets of arguments. Equality of individual arguments is defined as follows:

Argument Equality equality holds between any two arguments when (1) the role types are the same, and (2-1) both are t-entities and equality holds between them, or (2-2) both are events and equality holds between them.

Due to the condition (2-2), event equality is defined recursively for events referring to events. Equality of t-entities is defined as follows:

T-entity Equality equality holds between any two t-entities when (1) the entity types are the same, and (2) the spans are the same.

Any two text spans $(beg1, end1)$ and $(beg2, end2)$, are the same iff $beg1 = beg2$ and $end1 = end2$. Note that the event triggers are also t-entities thus their equality is defined by the t-entity equality.

4.2 Evaluation modes

Various evaluation modes can be defined by varying equivalence criteria. In the following, we describe three fundamental variants applied in the evaluation.

Strict matching The *strict matching* mode requires exact equality, as defined in section 4.1. As some of its requirements may be viewed as unnecessarily precise, practically motivated relaxed variants, described in the following, are also applied.

Approximate span matching The *approximate span matching* mode is defined by relaxing the requirement for text span matching for t-entities. Specifically, a given span is equivalent to a gold span if it is entirely contained within an extension of the gold span by one word both to the left and to the right, that is, $beg1 \geq ebeg2$ and $end1 \leq eend2$, where $(beg1, end1)$ is the given span and $(ebeg2, eend2)$ is the extended gold span.

Approximate recursive matching In strict matching, for a regulation event to be correct, the events it refers to as theme or cause must also be be strictly correct. The *approximate recursive matching* mode is defined by relaxing the requirement for recursive event matching, so that an event can match even if the events it refers to are only partially correct. Specifically, for partial matching, only Theme arguments are considered: events can match even if referred events differ in non-Theme arguments.

Event	Release date
Announcement	Dec 8
Sample data	Dec 15
Training data	Jan 19 → 21, Feb 2 (rev1), Feb 10 (rev2)
Devel. data	Feb 7
Test data	Feb 22 → Mar 2
Submission	Mar 2 → Mar 9

Table 3: Shared task schedule. The arrows indicate a change of schedule.

5 Schedule

The BioNLP task was held for 12 weeks, from the sample data release to the final submission. It included 5 weeks of *system design period* with sample data, 6 weeks of *system development period* with training and development data, and a 1 week *test period*. The system development period was originally planned for 5 weeks but extended by 1 week due to the delay of the training data release and the revision. Table 3 shows key dates of the schedule.

6 Supporting Resources

To allow participants to focus development efforts on novel aspects of event extraction, we prepared publicly available BioNLP resources readily available for the shared task. Several fundamental BioNLP tools were provided through U-Compare (Kano et al., 2009)[2], which included tools for tokenization, sentence segmentation, part-of-speech tagging, chunking and syntactic parsing.

Participants were also provided with the syntactic analyses created by a selection of parsers. We applied two mainstream Penn Treebank (PTB) phrase structure parsers: the Bikel parser[3], implementing Collins' parsing model (Bikel, 2004) and trained on PTB, and the reranking parser of (Charniak and Johnson, 2005) with the self-trained biomedical parsing model of (McClosky and Charniak, 2008)[4]. We also applied the GDep[5], native dependency parser trained on the GENIA Treebank

[2]http://u-compare.org/

[3]http://www.cis.upenn.edu/∼dbikel/software.html

[4]http://www.cs.brown.edu/∼dmcc/biomedical.html

[5]http://www.cs.cmu.edu/∼sagae/parser/gdep/

			NLP			Task		
Team	Task	Org	Word	Chunking	Parsing	Trigger	Argument	Ext. Resources
UTurku	`1--`	3C+2BI	Porter		MC	SVM	SVM (SVMlight)	
JULIELab	`1--`	1C+2L+2B	OpenNLP Porter	OpenNLP	GDep	Dict+Stat	SVM(libSVM) ME(Mallet)	UniProt, Mesh, GOA, UMLS
ConcordU	`1-3`	3C	Stanford		Stanford	Dict+Stat	Rules	WordNet, VerbNet, UMLS
UT+DBCLS	`12-`	2C	Porter		MC CCG	Dict	MLN(thebeast)	
VIBGhent	`1-3`	2C+1B	Porter,		Stanford	Dict	SVM(libSVM)	
UTokyo	`1--`	3C	GTag		GDep, Enju	Dict	ME(liblinear)	UIMA
UNSW	`1--`	1C+1B			GDep	CRF	Rules	WordNet, MetaMap
UZurich	`1--`	3C	LingPipe, Morpha	LTChunk	Pro3Gres	Dict	Rules	
ASU+HU+BU	`123`	6C+2BI	Porter		BioLG, Charniak	Dict	Rules Rules	Lucene
Cam	`1--`	3C	Porter		RASP	Dict	Rules	
UAntwerp	`12-`	3C	GTag		GDep	MBL	MBL(TiMBL) Rules	
UNIMAN	`1--`	4C+2BI	Porter GTag		GDep	Dict, CRF	SVM Rules	MeSH, GO
SCAI	`1--`	1C					Rules	
UAveiro	`1--`	1C+1L	NooJ	NooJ			Rules	BioLexicon
USzeged	`1-3`	3C+1B	GTag			Dict, VSM	C4.5(WEKA) Rules	BioScope
NICTA	`1-3`	4C	GTag		ERG	CRF(CRF++)	Rules	JULIE
CNBMadrid	`12-`	2C+1B	Porter, GTag	GTag			CBR Rules	
CCP-BTMG	`123`	7C	LingPipe	LingPipe	OpenDMAP	LingPipe, CM	Rules	GO, SO, MIO, UIMA
CIPS-ASU	`1--`	3C	MontyTagger	Custom	Stanford	CRF(ABNER)	Rules, NB(WEKA)	
UMich	`1--`	2C	Stanford		MC	Dict	SVM(SVMlight)	
PIKB	`1--`	5C+2B				MIRA	MIRA	
KoreaU	`1--`	5C	GTag		GDep	Rules, ME	ME	WSJ

Table 4: Profiles of the participants: GTag=GENIAtagger, MLN=Markov Logic Network, UMLS=UMLS SPECIALIST Lexicon/tools, MC=McClosky-Charniak, GDep=Genia Dependency Parser, Stanford=Stanford Parser, CBR=Case-Based Reasoning, CM=ConceptMapper.

(Tateisi et al., 2005), and a version of the C&C CCG deep parser[6] adapted to biomedical text (Rimell and Clark, 2008).

The text of all documents was segmented and tokenized using the GENIA Sentence Splitter and the GENIA Tagger, provided by U-Compare. The same segmentation was enforced for all parsers, which were run using default settings. Both the native output of each parser and a representation in the popular Stanford Dependency (SD) format (de Marneffe et al., 2006) were provided. The SD representation was created using the Stanford tools[7] to convert from the PTB scheme, the custom conversion introduced by (Rimell and Clark, 2008) for the C&C CCG parser, and a simple format-only conversion for GDep.

7 Results and Discussion

7.1 Participation

In total, 42 teams showed interest in the shared task and registered for participation, and 24 teams submitted final results. All 24 teams participated in the obligatory Task 1, six in each of Tasks 2 and 3, and two teams completed all the three tasks.

Table 4 shows a profile of the 22 final teams, excepting two who wished to remain anonymous. A brief examination on the team organization (the *Org* column) shows a computer science background (C) to be most frequent among participants, with less frequent participation from bioinformaticians (BI), biologists (B) and liguists (L). This may be attributed in part to the fact that the event extraction task required complex computational modeling. The role of computer scientists may be emphasized in part due to the fact that the task was novel to most participants, requiring particular efforts in framework design and implementation and computational resources. This also suggests there is room for improvement from more input from biologists.

7.2 Evaluation results

The final evaluation results of Task 1 are shown in Table 5. The results on the five event types involv-

[6]http://svn.ask.it.usyd.edu.au/trac/candc/wiki

[7]http://nlp.stanford.edu/software/lex-parser.shtml

Team	Simple Event	Binding	Regulation	All
UTurku	**64.21 / 77.45 / 70.21**	**40.06 / 49.82 / 44.41**	**35.63 / 45.87 / 40.11**	46.73 / 58.48 / 51.95
JULIELab	**59.81 / 79.80 / 68.38**	**49.57 / 35.25 / 41.20**	**35.03 / 34.18 / 34.60**	45.82 / 47.52 / 46.66
ConcordU	**49.75 / 81.44 / 61.76**	20.46 / 40.57 / 27.20	**27.47 / 49.89 / 35.43**	34.98 / 61.59 / 44.62
UT+DBCLS	**55.75 / 72.74 / 63.12**	23.05 / 48.19 / 31.19	**26.32 / 41.81 / 32.30**	36.90 / 55.59 / 44.35
VIBGhent	**54.48 / 79.31 / 64.59**	**38.04 / 38.60 / 38.32**	17.36 / 31.61 / 22.41	33.41 / 51.55 / 40.54
UTokyo	45.69 / 72.19 / 55.96	**34.58 / 50.63 / 41.10**	14.22 / 34.26 / 20.09	28.13 / 53.56 / 36.88
UNSW	45.85 / 69.94 / 55.39	23.63 / 37.27 / 28.92	16.58 / 28.27 / 20.90	28.22 / 45.78 / 34.92
UZurich	44.92 / 66.62 / 53.66	30.84 / 37.28 / 33.75	14.82 / 30.21 / 19.89	27.75 / 46.60 / 34.78
ASU+HU+BU	45.09 / 76.80 / 56.82	19.88 / 44.52 / 27.49	05.20 / 33.46 / 09.01	21.62 / 62.21 / 32.09
Cam	39.17 / 76.40 / 51.79	12.68 / 31.88 / 18.14	09.98 / 37.76 / 15.79	21.12 / 56.90 / 30.80
UAntwerp	41.29 / 65.68 / 50.70	12.97 / 31.03 / 18.29	11.07 / 29.85 / 16.15	22.50 / 47.70 / 30.58
UNIMAN	50.00 / 63.21 / 55.83	12.68 / 40.37 / 19.30	04.05 / 16.75 / 06.53	22.06 / 48.61 / 30.35
SCAI	43.74 / 70.73 / 54.05	28.82 / 35.21 / 31.70	12.64 / 16.55 / 14.33	25.96 / 36.26 / 30.26
UAveiro	43.57 / 71.63 / 54.18	13.54 / 34.06 / 19.38	06.29 / 21.05 / 09.69	20.93 / 49.30 / 29.38
Team 24	41.29 / 64.72 / 50.41	22.77 / 35.43 / 27.72	09.38 / 19.23 / 12.61	22.69 / 40.55 / 29.10
USzeged	47.63 / 44.44 / 45.98	15.27 / 25.73 / 19.17	04.17 / 18.21 / 06.79	21.53 / 36.99 / 27.21
NICTA	31.13 / 77.31 / 44.39	16.71 / 29.00 / 21.21	07.80 / 18.12 / 10.91	17.44 / 39.99 / 24.29
CNBMadrid	50.25 / 46.59 / 48.35	33.14 / 20.54 / 25.36	12.22 / 07.99 / 09.67	28.63 / 20.88 / 24.15
CCP-BTMG	28.17 / 87.63 / 42.64	12.68 / 40.00 / 19.26	03.09 / 48.11 / 05.80	13.45 / 71.81 / 22.66
CIPS-ASU	39.68 / 38.60 / 39.13	17.29 / 31.58 / 22.35	11.86 / 08.15 / 09.66	22.78 / 19.03 / 20.74
UMich	52.71 / 25.89 / 34.73	31.70 / 12.61 / 18.05	14.22 / 06.56 / 08.98	30.42 / 14.11 / 19.28
PIKB	26.65 / 75.72 / 39.42	07.20 / 39.68 / 12.20	01.09 / 30.51 / 02.10	11.25 / 66.54 / 19.25
Team 09	27.16 / 43.61 / 33.47	03.17 / 09.82 / 04.79	02.42 / 11.90 / 04.02	11.69 / 31.42 / 17.04
KoreaU	20.56 / 66.39 / 31.40	12.97 / 50.00 / 20.59	00.67 / 37.93 / 01.31	09.40 / 61.65 / 16.31

Table 5: Evaluation results of Task 1 (recall / precision / f-score).

Team	All	Site for Phospho.(56)	AtLoc & ToLoc (65)	All Second Args.
UT+DBCLS	**35.86 / 54.08 / 43.12**	**71.43 / 71.43 / 71.43**	**23.08 / 88.24 / 36.59**	**32.14 / 72.41 / 44.52**
UAntwerp	21.52 / 45.77 / 29.27	00.00 / 00.00 / 00.00	01.54 /100.00 / 03.03	06.63 / 52.00 / 11.76
ASU+HU+BU	19.70 / 56.87 / 29.26	00.00 / 00.00 / 00.00	00.00 / 00.00 / 00.00	00.00 / 00.00 / 00.00
Team 24	22.08 / 38.28 / 28.01	**55.36 / 93.94 / 69.66**	**21.54 / 66.67 / 32.56**	**30.10 / 76.62 / 43.22**
CCP-BTMG	13.25 / 70.97 / 22.33	30.36 /**100.00** / 46.58	00.00 / 00.00 / 00.00	08.67 /100.00 / 15.96
CNBMadrid	25.02 / 18.32 / 21.15	**85.71 / 57.14 / 68.57**	**32.31 / 47.73 / 38.53**	50.00 / 09.71 / 16.27

Table 6: Evaluation results for Task 2.

ing only a single primary theme argument are shown in one merged class, “Simple Event”. The broad performance range (31% – 70%) indicates even the extraction of simple events is not a trivial task. However, the top-ranked systems show encouraging performance, achieving or approaching 70% f-score.

The performance ranges for Binding (5% – 44%) and Regulation (1% – 40%) events show their extraction to be clearly more challenging. It is interesting that while most systems show better performance for binding over regulation events, the systems [ConcordU] and [UT+DBCLS] are better for regulation, showing somewhat reduced performance for Binding events. This is in particular contrast to the following two systems, [ViBGhent] and [UTokyo], which show far better performance for Binding than Regulation events. As one possible explanation, we find that the latter two differentiate binding events by their number of themes, while the former two give no specific treatment to multi-theme binding events. Such observations and comparisons are a clear benefit of a community-wide shared task.

Table 6 shows the evaluation results for the teams who participated in Task 2. The “All” column shows the overall performance of the systems for Task 2, while the “All Second Args.” column shows the performance of finding only the secondary arguments. The evaluation results show considerable differences between the criteria. For example, the system [Team 24] shows performance comparable to the top ranked system in finding secondary arguments, although its overall performance for Task 2 is more limited. Table 6 also shows the three systems, [UT+DBCLS], [Team 24] and [CNBMadrid],

Team	Negation	Speculation
ConcordU	**14.98 / 50.75 / 23.13**	**16.83 / 50.72 / 25.27**
VIBGhent	**10.57 / 45.10 / 17.13**	08.65 / 15.79 / 11.18
ASU+HU+BU	03.96 / 27.27 / 06.92	06.25 / 28.26 / 10.24
NICTA	05.29 / 34.48 / 09.17	04.81 / 30.30 / 08.30
USzeged	05.29 / 01.94 / 02.84	12.02 / 03.88 / 05.87
CCP-BTMG	01.76 / 05.26 / 02.64	06.73 / 13.33 / 08.95

Table 7: Evaluation results for Task 3.

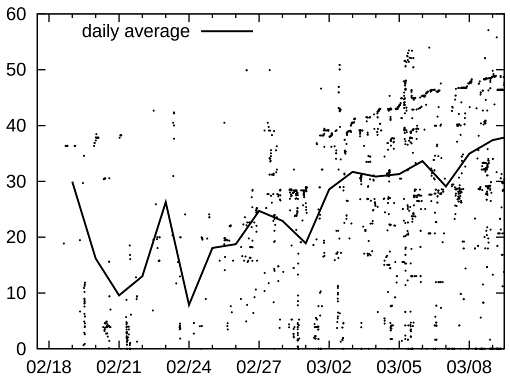


Figure 4: Scatterplot of the evaluation results on the development data during the system development period.

show performance at a practical level in particular in finding specific sites of phosphorylation.

As shown in Table 7, the performance range for Task 3 is very low although the representation of the task is as simple as the simple events. We attribute the reason to the fact that Task 3 is the only task of which the annotation is not bound to textual clue, thus no text-bound annotation was provided.

Figure 4 shows a scatter plot of the performance of the participating systems during the system development period. The performance evaluation comes from the log of the online evaluation system on the development data. It shows the best performance and the average performance of the participating systems were trending upwards up until the deadline of final submission, which indicates there is still much potential for improvement.

7.3 Ensemble

Table 8 shows experimental results of a system ensemble using the final submissions. For the experiments, the top 3–10 systems were chosen, and the output of each system treated as a weighted vote[8]. Three weighting schemes were used; "Equal" weights each vote equally; "Averaged" weights each vote by the overall f-score of the system; "Event Type" weights each vote by the f-score of the system for the specific event type. The best score, 55.96%, was obtained by the "Event Type" weighting scheme, showing a 4% unit improvement over the best individual system. While using the final scores for weighting uses data that would not be available in practice, similar weighting could likely be obtained e.g. using performance on the development data. The experiment demonstrates that an f-score better than 55% can be achieved simply by combining the strengths of the systems.

[8]We used the 'ensemble' function of U-Compare.

Ensemble	Equal	Averaged	Event Type
Top 3	53.19	53.19	54.08
Top 4	54.34	54.34	55.21
Top 5	54.77	55.03	55.10
Top 6	**55.13**	**55.77**	**55.96**
Top 7	54.33	55.45	55.73
Top 10	52.79	54.63	55.18

Table 8: Experimental results of system ensemble.

8 Conclusion

Meeting with the community-wide participation, the BioNLP Shared Task was successful in introducing fine-grained event extraction to the domain. The evaluation results of the final submissions from the participants are both promising and encouraging for the future of this approach to IE. It has been revealed that state-of-the-art performance in event extraction is approaching a practically applicable level for simple events, and also that there are many remaining challenges in the extraction of complex events. A brief analysis suggests that the submitted data together with the system descriptions are rich resources for finding directions for improvements. Finally, the experience of the shared task participants provides an invaluable basis for cooperation in facing further challenges.

Acknowledgments

This work was partially supported by Grant-in-Aid for Specially Promoted Research (MEXT, Japan) and Genome Network Project (MEXT, Japan).

References

Gary D. Bader, Michael P. Cary, and Chris Sander. 2006. Pathguide: a Pathway Resource List. *Nucleic Acids Research.*, 34(suppl_1):D504–506.

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

Evelyn Camon, Michele Magrane, Daniel Barrell, Vivian Lee, Emily Dimmer, John Maslen, David Binns, Nicola Harte, Rodrigo Lopez, and Rolf Apweiler. 2004. The Gene Ontology Annotation (GOA) Database: sharing knowledge in Uniprot with Gene Ontology. *Nucl. Acids Res.*, 32(suppl 1):D262–266.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-Fine n-Best Parsing and MaxEnt Discriminative Reranking. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 173–180.

Andrew Chatr-aryamontri, Arnaud Ceol, Luisa Montecchi Palazzi, Giuliano Nardelli, Maria Victoria Schneider, Luisa Castagnoli, and Gianni Cesareni. 2007. MINT: the Molecular INTeraction database. *Nucleic Acids Research*, 35(suppl 1):D572–574.

Nancy Chinchor. 1998. Overview of MUC-7/MET-2. In *Message Understanding Conference (MUC-7) Proceedings*.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating Typed Dependency Parses from Phrase Structure Parses. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC'06)*, pages 449–454.

William Hersh, Aaron Cohen, Ruslenm Lynn, , and Phoebe Roberts. 2007. TREC 2007 Genomics track overview. In *Proceeding of the Sixteenth Text REtrieval Conference*.

Lynette Hirschman, Martin Krallinger, and Alfonso Valencia, editors. 2007. *Proceedings of the Second BioCreative Challenge Evaluation Workshop*. CNIO Centro Nacional de Investigaciones Oncológicas.

Yoshinobu Kano, William Baumgartner, Luke McCrohon, Sophia Ananiadou, Kevin Cohen, Larry Hunter, and Jun'ichi Tsujii. 2009. U-Compare: share and compare text mining tools with UIMA. *Bioinformatics*. To appear.

Jin-Dong Kim, Tomoko Ohta, Yoshimasa Tsuruoka, Yuka Tateisi, and Nigel Collier. 2004. Introduction to the bio-entity recognition task at JNLPBA. In *Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and its Applications (JNLPBA)*, pages 70–75.

Jin-Dong Kim, Tomoko Ohta, and Jun'ichi Tsujii. 2008. Corpus annotation for mining biomedical events from lterature. *BMC Bioinformatics*, 9(1):10.

David McClosky and Eugene Charniak. 2008. Self-Training for Biomedical Parsing. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics - Human Language Technologies (ACL-HLT'08)*, pages 101–104.

Claire Nédellec. 2005. Learning Language in Logic - Genic Interaction Extraction Challenge. In J. Cussens and C. Nédellec, editors, *Proceedings of the 4th Learning Language in Logic Workshop (LLL05)*, pages 31–37.

Tomoko Ohta, Jin-Dong Kim, Sampo Pyysalo, and Jun'ichi Tsujii. 2009. Incorporating GENETAG-style annotation to GENIA corpus. In *Proceedings of Natural Language Processing in Biomedicine (BioNLP) NAACL 2009 Workshop*. To appear.

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

Sampo Pyysalo, Tomoko Ohta, Jin-Dong Kim, and Jun'ichi Tsujii. 2009. Static Relations: a Piece in the Biomedical Information Extraction Puzzle. In *Proceedings of Natural Language Processing in Biomedicine (BioNLP) NAACL 2009 Workshop*. To appear.

Laura Rimell and Stephen Clark. 2008. Porting a lexicalized-grammar parser to the biomedical domain. *Journal of Biomedical Informatics*, To Appear.

Stephanie Strassel, Mark Przybocki, Kay Peterson, Zhiyi Song, and Kazuaki Maeda. 2008. Linguistic Resources and Evaluation Techniques for Evaluation of Cross-Document Automatic Content Extraction. In *Proceedings of the 6th International Conference on Language Resources and Evaluation (LREC 2008)*.

Lorraine Tanabe, Natalie Xie, Lynne Thom, Wayne Matten, and John Wilbur. 2005. Genetag: a tagged corpus for gene/protein named entity recognition. *BMC Bioinformatics*, 6(Suppl 1):S3.

Yuka Tateisi, Akane Yakushiji, Tomoko Ohta, and Jun'ichi Tsujii. 2005. Syntax Annotation for the GENIA corpus. In *Proceedings of the IJCNLP 2005, Companion volume*, pages 222–227.

Ellen Voorhees. 2007. Overview of TREC 2007. In *The Sixteenth Text REtrieval Conference (TREC 2007) Proceedings*.

John Wilbur, Lawrence Smith, and Lorraine Tanabe. 2007. BioCreative 2. Gene Mention Task. In L. Hirschman, M. Krallinger, and A. Valencia, editors, *Proceedings of Second BioCreative Challenge Evaluation Workshop*, pages 7–16.

Extracting Complex Biological Events with Rich Graph-Based Feature Sets

Jari Björne,[1] **Juho Heimonen**,[1,2] **Filip Ginter**,[1] **Antti Airola**,[1,2]
Tapio Pahikkala[1] and **Tapio Salakoski**[1,2]
[1]Department of Information Technology, University of Turku
[2]Turku Centre for Computer Science (TUCS)
Joukahaisenkatu 3-5, 20520 Turku, Finland
firstname.lastname@utu.fi

Abstract

We describe a system for extracting complex events among genes and proteins from biomedical literature, developed in context of the BioNLP'09 Shared Task on Event Extraction. For each event, its text trigger, class, and arguments are extracted. In contrast to the prevailing approaches in the domain, events can be arguments of other events, resulting in a nested structure that better captures the underlying biological statements. We divide the task into independent steps which we approach as machine learning problems. We define a wide array of features and in particular make extensive use of dependency parse graphs. A rule-based post-processing step is used to refine the output in accordance with the restrictions of the extraction task. In the shared task evaluation, the system achieved an F-score of 51.95% on the primary task, the best performance among the participants.

1 Introduction

In this paper, we present the best-performing system in the primary task of the BioNLP'09 Shared Task on Event Extraction (Kim et al., 2009).[1] The purpose of this shared task was to competitively evaluate information extraction systems targeting complex events in the biomedical domain. Such an evaluation helps to establish the relative merits of competing approaches, allowing direct comparability of results in a controlled setting. The shared task was the first competitive evaluation of its kind in the BioNLP field as the extraction of complex events became possible only recently with the introduction of corpora containing the necessary annotation: the GENIA event corpus (Kim et al., 2008a) and the BioInfer corpus (Pyysalo et al., 2007).

The objective of the primary task (Task 1) was to detect biologically relevant events such as protein binding and phosphorylation, given only annotation of named entities. For each event, its class, trigger expression in the text, and arguments need to be extracted. The task follows the recent movement in BioNLP towards the extraction of semantically typed, complex events the arguments of which can also be other events. This results in a nested structure that captures the underlying biological statements more accurately compared to the prevailing approach of merely detecting binary interactions of pairs of biological entities.

Our system is characterized by heavy reliance on efficient, state-of-the-art machine learning techniques and a wide array of features derived from a full dependency analysis of each sentence. The system is a pipeline of three major processing steps: trigger recognition, argument detection and semantic post-processing. By separating trigger recognition from argument detection, we can use methods familiar from named entity recognition to tag words as event triggers. Event argument detection then becomes the task of predicting for each trigger–trigger or trigger–named entity pair whether it corresponds to an actual instantiation of an event argument. Both steps can thus be approached as classification tasks. In contrast, semantic post-processing

[1]http://www-tsujii.is.s.u-tokyo.ac.jp/GENIA/SharedTask

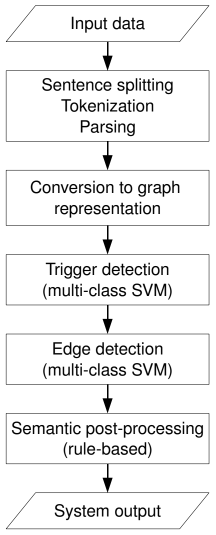

Input data
Sentence splitting
Tokenization
Parsing
Conversion to graph representation
Trigger detection (multi-class SVM)
Edge detection (multi-class SVM)
Semantic post-processing (rule-based)
System output

Figure 1: The main components of the system.

is rule-based, directly implementing argument type constraints following from the definition of the task.

In the following sections, we present the implementation of the three stages of our information extraction system in detail, and provide insights into why we chose the approach we did. We also discuss alternate directions we followed but that did not improve performance. Finally, we analyze the overall performance of our system in the shared task as well as evaluate its components individually.

2 The system description

The overall architecture of the system is shown in Figure 1. All steps in the system process one sentence at a time. Since 95% of all annotated events are fully contained within a single sentence, this does not incur a large performance penalty but greatly reduces the size and complexity of the machine learning problems.

2.1 Graph representation

We represent the extraction target in terms of semantic networks, graphs where the nodes correspond to named entities and events, and the edges correspond to event arguments. The shared task can then be viewed as the problem of finding the nodes and edges of this graph. For instance, nested events are naturally represented through edges connecting two event nodes. The graph representation of an example sentence is illustrated in Figure 2D.

We have previously used this graph representation for information extraction (Heimonen et al., 2008; Björne et al., 2009) as well as for establishing the connection between events and syntactic dependency parses in the Stanford scheme of de Marneffe and Manning (2008) (Björne et al., 2008).

2.2 Trigger detection

We cast trigger detection as a token labeling problem, that is, each token is assigned to an event class, or a negative class if it does not belong to a trigger. Triggers are then formed based on the predicted classes of the individual tokens. Since 92% of all triggers in the data consist of a single token, adjacent tokens with the same class prediction form a single trigger only in case that the resulting string occurs as a trigger in the training data. An event node is created for each detected trigger (Figure 2B).

In rare cases, the triggers of events of different class share a token, thus the token belongs to several separate classes. To be able to approach trigger detection as a multi-class classification task where each token is given a single prediction, we introduce combined classes as needed. For instance the class *gene expression/positive regulation* denotes tokens that act as a trigger to two events of the two respective classes. Note that this implies that the trigger detection step produces at most one event node per class for any detected trigger. In the shared task, however, multiple events of the same class can share the same trigger. For instance, the trigger *involves* in Figure 2 corresponds to two separate *regulation* events. A separate post-processing step is introduced after event argument detection to duplicate event nodes as necessary (see Section 2.4).

Due to the nature of the GENIA event annotation principles, trigger detection cannot be easily reduced to a simple dictionary lookup of trigger expressions for two main reasons. First, a number of common textual expressions act as event triggers in some cases, but not in other cases. For example, only 28% of the instances of the expression *activates* are triggers for a *positive regulation* event while the remaining 72% are not triggers for any event. Second, a single expression may be associated with various event classes. For example, the instances of the token *overexpression* are evenly distributed among

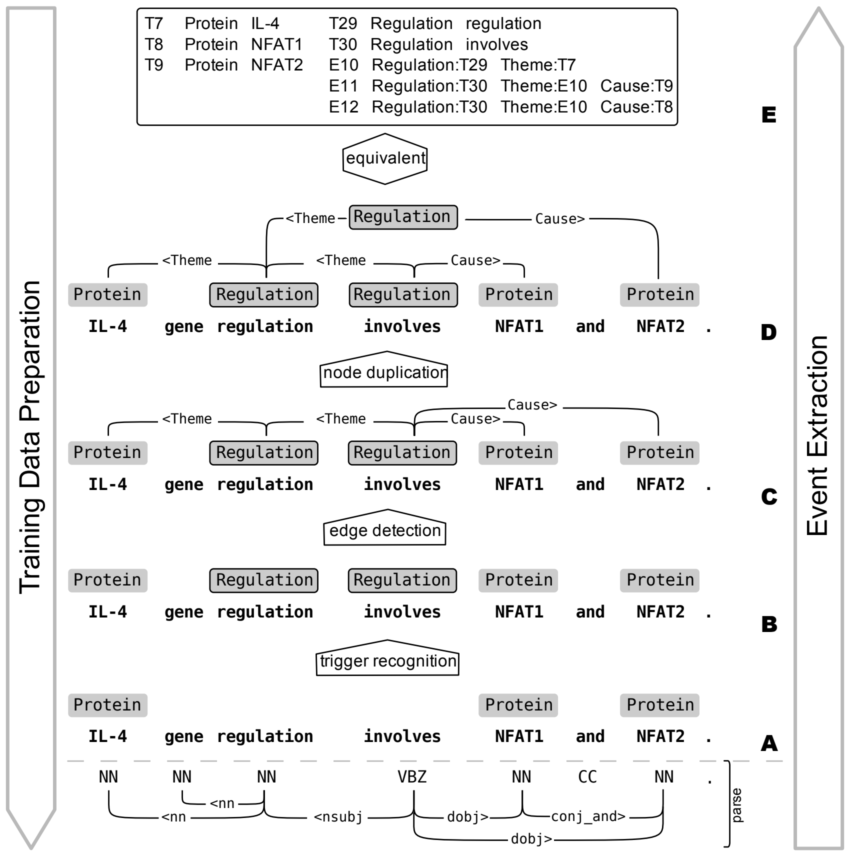


Figure 2: An example sentence from Shared Task document 10069428 (simplified). A) Named entities are given. B) Triggers are detected and corresponding event nodes are created. C) Event argument edges are predicted between nodes. The result is a sentence-level semantic network. D) One node may denote multiple events of the same class, therefore nodes are duplicated in the semantic post-processing step. E) The resulting graph can be losslessly transformed into the Shared Task event annotation. Training data for the trigger recognizer includes named entity annotation (A) and for the edge detector the semantic network with no node duplication (C).

gene expression, *positive regulation*, and the negative class. In light of these properties, we address trigger detection with a multi-class support vector machine (SVM) classifier that assigns event classes to individual tokens, one at a time. This is in contrast to sequence labeling problems such as named entity recognition, where a sequential model is typically employed. The classifier is trained on gold-standard triggers from the training data and incorporates a wide array of features capturing the properties of the token to be classified, both its linear and dependency context, and the named entities within the sentence.

Token features include binary tests for capitalization, presence of punctuation or numeric characters, stem using the Porter stemmer (Porter, 1980), character bigrams and trigrams, and presence of the token in a gazetteer of known trigger expressions and their classes, extracted from the training data. Token features are generated not only for the token to be classified, but also for tokens in the immediate linear context and dependency context (tokens that govern or depend on the token to be classified).

Frequency features include the number of named entities in the sentence and in a linear window around the token in question as well as bag-of-word counts of token texts in the sentence.

Dependency chains up to depth of three are constructed, starting from the token to be classified. At each depth, both token features and dependency type are included, as well as the sequence of dependency types in the chain.

The trigger detector used in the shared task is in fact a weighted combination of two independent SVM trigger detectors, both based on the same multi-class classification principle and somewhat different feature sets.[2] The predictions of the two trigger detectors are combined as follows. For each trigger detector and each token, the classifier confidence scores of the top five classes are re-normalized into the $[0, 1]$ interval. The renormalized confidence scores of the two detectors are then linearly interpolated using a parameter λ, $0 \leq \lambda \leq 1$, whose value is set experimentally on the development set, as discussed below.

Setting the correct precision–recall trade-off in trigger detection is very important. On one hand, any trigger left undetected directly implies a false negative event. On the other hand, the edge detector is trained on gold standard data where there are no event nodes without arguments, which creates a bias toward predicting edges for any event node the edge detector is presented with. On the development set, essentially all predicted event nodes are given at least one argument edge. We optimize the precision–recall trade-off explicitly by introducing a parameter β, $0 \leq \beta$, that multiplies the classifier confidence score given to the negative class, that is, the "no trigger" class. When $\beta < 1$, the confidence of the negative class is decreased, thus increasing the possibility of a given token forming a trigger, and consequently increasing the recall of the trigger detector (naturally, at the expense of its precision).

Both trigger detection parameters, the interpolation weight λ and the precision–recall trade-off parameter β, are set experimentally using a grid search to find the globally optimal performance of the entire system on the development set, using the shared task performance metric. The parameters are thus not set to optimize the performance of trigger detection in isolation; they are rather set to optimize the performance of the whole system.

2.3 Edge detection

After trigger detection, edge detection is used to predict the edges of the semantic graph, thus extracting event arguments. Like the trigger detector, the edge detector is based on a multi-class SVM classifier. We generate examples for all potential edges, which

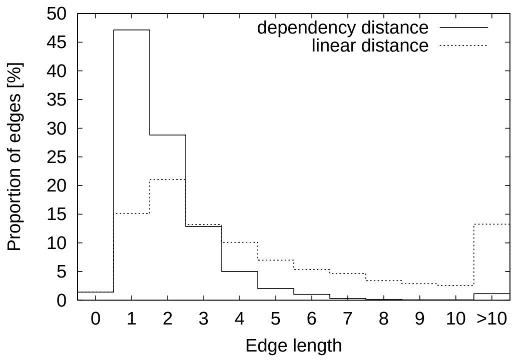


Figure 3: The distribution of event argument edge lengths measured as the number of dependencies on the shortest dependency path between the edge terminal nodes, contrasted with edge lengths measured as the linear token distance.

are always directed from an event node to another event node (event nesting) or from an event node to a named entity node. Each example is then classified as *theme*, *cause*, or a negative denoting the absence of an edge between the two nodes in the given direction. It should be noted that even though event nodes often require multiple outgoing edges corresponding to multiple event arguments, all edges are predicted independently and are not affected by positive or negative classifications of other edges.

The feature set makes extensive use of syntactic dependencies, in line with many recent studies in biomedical information extraction (see, e.g. (Kim et al., 2008b; Miwa et al., 2008; Airola et al., 2008; Van Landeghem et al., 2008; Katrenko and Adriaans, 2008)). The central concept in generating features of potential event argument edges is the *shortest undirected path of syntactic dependencies* in the Stanford scheme parse of the sentence which we assume to accurately capture the relationship expressed by the edge. In Figure 3, we show that the distances among event and named entity nodes in terms of shortest dependency path length are considerably shorter than in terms of their linear order in the sentence. The end points of the path are the syntactic head tokens of the two named entities or event triggers. The head tokens are identified using a simple heuristic. Where multiple shortest paths exist, all are considered. Most features are built by combining the attributes of multiple tokens (token text,

[2]This design should be considered an artifact of the time-constrained, experiment-driven development of the system rather than a principled design choice.

POS tag and entity or event class, such as *protein* or *binding*) or dependencies (type such as *subject* and direction relative to surrounding tokens).

***N*-grams** are generated by merging the attributes of 2–4 consecutive tokens. *N*-grams are also built for consecutive dependencies. Additional trigrams are built for each token and its two flanking dependencies, as well as for each dependency and its two flanking tokens. These *N*-grams are defined in the direction of the potential event argument edge. To take into account the varying directions of the dependencies, each pair of consecutive tokens forms an additional bigram defining their governor-dependent relationship.

Individual component features are defined for each token and edge in a path based on their attributes which are also combined with the token/edge position at either the interior or the end of the path. Edge attributes are combined with their direction relative to the path.

Semantic node features are built by directly combining the attributes of the two terminal event/entity nodes of the potential event argument edge. These features concatenate both the specific types of the nodes (e.g. *protein* or *binding*) as well as their categories (event or named entity). Finally, if the events/entities have the same head token, this self-loop is explicitly defined as a feature.

Frequency features include the length of the shortest path as an integer-valued feature as well as an explicit binary feature for each length. The number of named entities and event nodes, per type, in the sentence are defined for each example.

We have used this type of edge detector with a largely similar feature set previously (Björne et al., 2009). Also, many of these features are standard in relation extraction studies (see, e.g., Buyko et al. (2008)).

2.4 Semantic post-processing

The semantic graph produced by the trigger and edge detection steps is not final. In particular, it may contain event nodes with an improper combination of arguments, or no arguments whatsoever. Additionally, as discussed in Section 2.2, if there are events of the same class with the same trigger, they are represented by a single node. Therefore, we introduce a rule-based post-processing step to refine

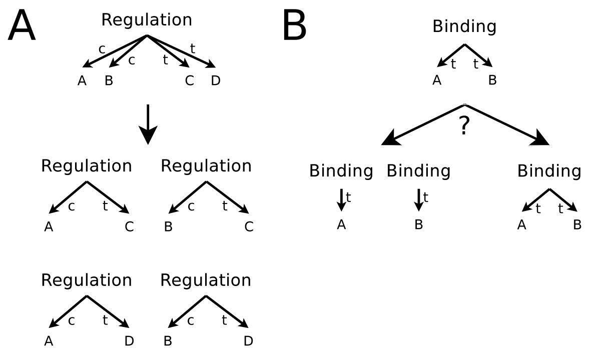


Figure 4: Example of event duplication. A) All *theme–cause* combinations are generated for *regulation* events. B) A heuristic is applied to decide how *theme* arguments of *binding* events should be grouped.

the graph, using the restrictions on event argument types and combinations defined in the shared task.

In Task 1, the allowed argument edges in the graph are 1) *theme* from an event to a named entity, 2) *theme* or *cause* from a *regulation* event (or its subclasses) to an event or a named entity. Edges corresponding to invalid arguments are removed. Also, directed cycles are broken by removing the edge with the weakest classification confidence score.

After pruning invalid edges, event nodes are duplicated so that all events have a valid combination of arguments. For example, the *regulation* event *involves* in Figure 2C has two *cause* arguments and therefore represents two distinct events. We thus duplicate the event node, obtaining one *regulation* event for each of the *cause* arguments (Figure 2D).

Events of type *gene expression*, *transcription*, *translation*, *protein catabolism*, *localization*, and *phosphorylation* must have exactly one *theme* argument, which makes the duplication process trivial: duplicate events are created, one for each of the arguments. *Regulation* events must have one *theme* and can additionally have one *cause* argument. For these classes we use a heuristic, generating a new event for each *theme–cause* combination of outgoing edges (Figure 4A). *Binding* is the only event class that can have multiple *theme* arguments. There is thus no simple way of determining how multiple outgoing *theme* edges should be grouped (Figure 4B). We apply a heuristic that first groups the arguments by their syntactic role, defined here as xthe first dependency in the shortest path from the event to the argument. It then generates an event for each pair of arguments that are in different groups. In the case of only one group, all single-argument events are generated.

Finally, all events with no arguments as well as *regulation* events without a *theme* argument are iteratively removed until no such event remains. The resulting graph is the output of our event extraction system and can be losslessly converted into the shared task format (Figure 2D&E).

2.5 Alternative directions

We now briefly describe some of the alternative directions explored during the system development, which however did not result in increased performance, and were thus not included in the final system. Whether the reason was due to the considered approaches being inadequate for the extraction task, or simply a result of the tight time constraints enforced by the shared task is a question only further research can shed light on.

For the purpose of dividing the extraction problem into manageable subproblems, we make strong independence assumptions. This is particularly the case in the edge detection phase where each edge is considered in isolation from other edges, some of which may actually be associated with the same event. Similar assumptions are made in the trigger detection phase, where the classifications of individual tokens are independent.

A common way to relax independence assumptions is to use N-best re-ranking where N most-likely candidates are re-ranked using global features that model data dependencies that could not be modelled in the candidate generation step. The best candidate with respect to this re-ranked order is then the final prediction of the system. N-best re-ranking has been successfully applied for example in statistical parsing (Charniak and Johnson, 2005). We generated the ten most likely candidate graphs, as determined by the confidence scores of the individual edges given by the multi-class SVM. A perfect re-ranking of these ten candidates would lead to 11.5 percentage point improvement in the overall system F-score on the development set. While we were unable to produce a re-ranker sufficiently accurate to improve the system performance in the time given, the large potential gain warrants further research.

In trigger word detection, we experimented with a structural SVM incorporating Hidden Markov Model type of sequential dependencies (Altun et al., 2003; Tsochantaridis et al., 2004), which allow conditioning classification decisions on decisions made for previous tokens as well as with a conditional random field (CRF) sequence classifier (Lafferty et al., 2001). Neither of these experiments led to a performance gain over the multiclass SVM classifier.

As discussed previously, 4.8% of all annotated events cross sentence boundaries. This problem could be approached using coreference resolution techniques, however, the necessary explicit coreference annotation to train a coreference resolution system is not present in the data. Instead, we attempted to build a machine-learning based system to detect cross-sentence event arguments directly, rather than via their referring expression, but were unable to improve the system performance.

3 Tools and resources

3.1 Multi-class SVM

We use a support vector machine (SVM) multi-class classifier which has been shown to have state-of-the-art classification performance (see e.g. (Crammer and Singer, 2002; Tsochantaridis et al., 2004)). Namely, we use the SVM*multiclass* implementation[3] which is one of the fastest multi-class SVM implementations currently available. Analogously to the binary SVMs, multi-class SVMs have a regularization parameter that determines the trade-off between the training error and the complexity of the learned concept. We select the value of the parameter on the development set. Multi-class SVMs scale linearly with respect to both the amount of training data and the average number of nonzero features per training example, making them an especially suitable learning method for our purposes. They also provide a real-valued prediction for each example to be classified which is used as a confidence score in trigger detection precision–recall trade-off adjustment and event argument edge cycle breaking in semantic post-processing. We use the linear kernel, the only practical choice to train the classifier with the large training sets available. For example, the

[3]`http://svmlight.joachims.org/svm_multiclass.html`

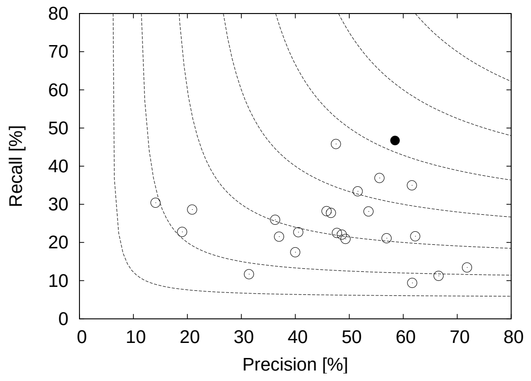


Figure 5: Performance of the 24 systems that participated in Task 1, together with an F-score contour plot for reference. Our system is marked with a full circle.

final training data of the edge detector (8932 sentences) consists of 31792 training examples with 295034 unique features. Training with even this amount of data is computationally feasible, typically taking less than an hour.

All classifiers used in the system are trained as follows. First we optimize the regularization parameter C by training on the shared task training set and testing on the shared task development set. We then re-train the final classifier on the union of the training and development sets, using the best value of C in the previous step. The same protocol is followed for the λ and β parameters in trigger detection.

3.2 Dependency parses

Both trigger detection and edge prediction rely on a wide array of features derived from full dependency parses of the sentence. We use the McClosky-Charniak domain-adapted parser (McClosky and Charniak, 2008) which is among the best performing parsers trained on the GENIA Treebank corpus. The native constituency output of the parser is transformed to the "collapsed" form of the Stanford dependency scheme (de Marneffe and Manning, 2008) using the Stanford parser tools.[4] The parses were provided by the shared task organizers.

4 Results and discussion

The final evaluation of the system was performed by the shared task organizers using a test set whose annotation was at no point available to the task participants. By the main criterion of Task 1, approximate span matching with approximate recursive matching, our system achieved an F-score of 51.95%. Figure 5 shows the performance of all systems participating in Task 1. The per-class results in Table 1 show that *regulation* events (including *positive* and *negative regulation*) as well as *binding* events are the hardest to extract. These classes have F-scores in the 31–44% range, while the other classes fall into the 50–78% range. This is not particularly surprising since *binding* and *regulation* are the only classes in which events can have multiple arguments, which means that for an event to be detected correctly, the edge detector often must make several correct predictions. Additionally, these classes have the lowest trigger recognition performance on the development set. It is interesting to note that the per-class performance in Table 1 shows no clear correlation between the number of events of a class and its F-score.

Table 2 shows the performance of the system using various other evaluation criteria defined in the shared task. The most interesting of these is the *strict* matching criterion, which, in order to consider an event correctly extracted, requires exact trigger span as well as all its nested events to be recursively correct. The performance of the system with respect to the strict criterion is 47.41% F-score, only 4.5 percentage points lower than the relaxed primary measure. As seen in Table 2, this difference is almost exclusively due to triggers with incorrect span.

To evaluate the performance impact of each system component individually, we report in Table 3 overall system performance on the development set, obtained by progressively replacing the processing steps with gold-standard data. The results show that the errors of the system are almost evenly distributed between the trigger and edge detectors. For instance, a perfect trigger detector would decrease the overall system error of 46.5% by 18.58 percentage points, a relative decrease of 40%. A perfect edge detector would, in combination with a perfect trigger detector, lead to system performance of 94.69%. The improvement that could be gained by further development of the semantic post-processing step is thus limited, indicating that the strict argument combination restrictions of Task 1 are sufficient to resolve the majority of post-processing cases.

[4]`http://nlp.stanford.edu/software/`

Event Class	#	R	P	F
Protein catabolism	14	42.86	66.67	52.17
Phosphorylation	135	80.74	74.66	77.58
Transcription	137	39.42	69.23	50.23
Localization	174	49.43	81.90	61.65
Regulation	291	25.43	38.14	30.52
Binding	347	40.06	49.82	44.41
Negative regulation	379	35.36	43.46	38.99
Gene expression	722	69.81	78.50	73.90
Positive regulation	983	38.76	48.72	43.17
Total	3182	46.73	58.48	51.95

Table 1: Per-class performance in terms of Recall, Precision, and F-score on the test set (3182 events) using approximate span and recursive matching, the primary evaluation criterion of Task 1.

Matching	R	P	F
Strict	42.65	53.38	47.41
Approx. Span	46.51	58.24	51.72
Approx. Span&Recursive	46.73	58.48	51.95

Table 2: Performance of our system on the test set (3182 events) with respect to other evaluation measures in the shared task.

5 Conclusions

We have described a system for extracting complex, typed events from biomedical literature, only assuming named entities as given knowledge. The high rank achieved in the BioNLP'09 Shared Task competitive evaluation validates the approach taken in building the system. While the performance is currently the highest achieved on this data, the F-score of 51.95% indicates that there remains considerable room for further development and improvement.

We use a unified graph representation of the data in which the individual processing steps can be formulated as simple graph transformations: adding or removing nodes and edges. It is our experience that such a representation makes handling the data fast, easy and consistent. The choice of graph representation is further motivated by the close correlation of these graphs with dependency parses. As we are going to explore the interpretation and applications of these graphs in the future, the graph representation will likely provide a flexible base to build on.

Dividing the task of event extraction into multiple subtasks that can be approached by well-studied methods proved to be an effective approach in developing our system. We relied on state-of-the-art machine learning techniques that scale up to the task and allow the use of a considerable number of features. We also carefully optimized the various parameters, a vital step when using machine learning methods, to fine-tune the performance of the system.

Trig	Edge	PP	R	P	F	ΔF
pred	pred	pred	51.54	55.62	53.50	
GS	pred	pred	71.66	72.51	72.08	18.58
GS	GS	pred	97.21	92.30	94.69	22.61
GS	GS	GS	100.0	100.0	100.0	5.31

Table 3: Effect of the trigger detector (*Trig*), edge detector (*Edge*), and post-processing (*PP*) on performance on the development set (1789 events). The ΔF column indicates the effect of replacing the predictions (*pred*) of a component with the corresponding gold standard data (*GS*), i.e. the maximal possible performance gain obtainable from further development of that component.

In Section 2.5, we discussed alternative directions pursued during the development of the current system, indicating possible future research directions. To support this future work as well as complement the description of the system in this paper we intend to publish our system under an open-source license.

This shared task represents the first competitive evaluation of complex event extraction in the biomedical domain. The prior research has largely focused on binary interaction extraction, achieving after a substantial research effort F-scores of slightly over 60% (see, e.g., Miwa et al. (2008)) on AIMed, the *de facto* standard corpus for this task. Even if a direct comparison of these results is difficult, they suggest that 52% F-score in complex event extraction is a non-trivial achievement, especially considering the more detailed semantics of the extracted events. Further, complex event extraction is still a new problem — relevant corpora having been available for only a few years.

Acknowledgments

This research was funded by the Academy of Finland. Computational resources were provided by CSC — IT Center for Science Ltd. We thank the shared task organizers for their efforts in data preparation and system evaluation.

References

Antti Airola, Sampo Pyysalo, Jari Björne, Tapio Pahikkala, Filip Ginter, and Tapio Salakoski. 2008. All-paths graph kernel for protein-protein interaction extraction with evaluation of cross-corpus learning. *BMC Bioinformatics*, 9(Suppl 11):S2.

Yasemin Altun, Ioannis Tsochantaridis, and Thomas Hofmann. 2003. Hidden Markov support vector machines. In *Proceedings of the Twentieth International Conference on Machine Learning (ICML'03)*, pages 3–10. AAAI Press.

Jari Björne, Sampo Pyysalo, Filip Ginter, and Tapio Salakoski. 2008. How complex are complex protein-protein interactions? In *Proceedings of the Third International Symposium on Semantic Mining in Biomedicine (SMBM'08)*, pages 125–128. TUCS.

Jari Björne, Filip Ginter, Juho Heimonen, Sampo Pyysalo, and Tapio Salakoski. 2009. Learning to extract biological event and relation graphs. In *Proceedings of the 17th Nordic Conference on Computational Linguistics (NODALIDA'09)*.

Ekaterina Buyko, Elena Beisswanger, and Udo Hahn. 2008. Testing different ACE-style feature sets for the extraction of gene regulation relations from MEDLINE abstracts. In *Proceedings of the Third International Symposium on Semantic Mining in Biomedicine (SMBM'08)*, pages 21–28. TUCS.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 173–180. ACL.

Koby Crammer and Yoram Singer. 2002. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, 2:265–292.

Marie-Catherine de Marneffe and Christopher Manning. 2008. Stanford typed hierarchies representation. In *Proceedings of the COLING'08 Workshop on Cross-Framework and Cross-Domain Parser Evaluation*, pages 1–8.

Juho Heimonen, Sampo Pyysalo, Filip Ginter, and Tapio Salakoski. 2008. Complex-to-pairwise mapping of biological relationships using a semantic network representation. In *Proceedings of the Third International Symposium on Semantic Mining in Biomedicine (SMBM'08)*, pages 45–52. TUCS.

Sophia Katrenko and Pieter Adriaans. 2008. A local alignment kernel in the context of NLP. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling'08)*.

Jin-Dong Kim, Tomoko Ohta, and Tsujii Jun'ichi. 2008a. Corpus annotation for mining biomedical events from literature. *BMC Bioinformatics*, 9(1):10.

Seonho Kim, Juntae Yoon, and Jihoon Yang. 2008b. Kernel approaches for genic interaction extraction. *Bioinformatics*, 24(1):118–126.

Jin-Dong Kim, Tomoko Ohta, Sampo Pyysalo, Yoshinobu Kano, and Jun'ichi Tsujii. 2009. Overview of BioNLP'09 shared task on event extraction. In *Proceedings of the NAACL-HLT 2009 Workshop on Natural Language Processing in Biomedicine (BioNLP'09)*. ACL.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning (ICML'01)*, pages 282–289.

David McClosky and Eugene Charniak. 2008. Self-training for biomedical parsing. In *Proceedings of ACL-08: HLT, Short Papers*, pages 101–104. Association for Computational Linguistics.

Makoto Miwa, Rune Sætre, Yusuke Miyao, Tomoko Ohta, and Jun'ichi Tsujii. 2008. Combining multiple layers of syntactic information for protein-protein interaction extraction. In *Proceedings of the Third International Symposium on Semantic Mining in Biomedicine (SMBM'08)*, pages 101–108. TUCS.

Martin F. Porter. 1980. An algorithm for suffix stripping. *Program*, 14(3):130–137.

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

Ioannis Tsochantaridis, Thomas Hofmann, Thorsten Joachims, and Yasemin Altun. 2004. Support vector machine learning for interdependent and structured output spaces. In *Proceedings of the Twenty-first International Conference on Machine Learning (ICML'04)*, pages 104–111. ACM.

Sofie Van Landeghem, Yvan Saeys, Bernard De Baets, and Yves Van de Peer. 2008. Extracting protein-protein interactions from text using rich feature vectors and feature selection. In *Proceedings of the Third International Symposium on Semantic Mining in Biomedicine (SMBM'08)*, pages 77–84. TUCS.

Event Extraction from Trimmed Dependency Graphs

Ekaterina Buyko, Erik Faessler, Joachim Wermter and **Udo Hahn**
Jena University Language & Information Engineering (JULIE) Lab
Friedrich-Schiller-Universität Jena
Fürstengraben 30, 07743 Jena, Germany
{ekaterina.buyko|erik.faessler|joachim.wermter|udo.hahn}@uni-jena.de

Abstract

We describe the approach to event extraction which the JULIELab Team from FSU Jena (Germany) pursued to solve Task 1 in the "BioNLP'09 Shared Task on Event Extraction". We incorporate manually curated dictionaries and machine learning methodologies to sort out associated event triggers and arguments on trimmed dependency graph structures. Trimming combines pruning irrelevant lexical material from a dependency graph and decorating particularly relevant lexical material from that graph with more abstract conceptual class information. Given that methodological framework, the JULIELab Team scored on 2nd rank among 24 competing teams, with 45.8% precision, 47.5% recall and 46.7% F1-score on all 3,182 events.

1 Introduction

Semantic forms of text analytics for the life sciences have long been equivalent with named entity recognition and interpretation, i.e., finding instances of semantic classes such as proteins, diseases, or drugs. For a couple of years, this focus has been complemented by analytics dealing with relation extraction, i.e., finding instances of relations which link one or more (usually two) arguments, the latter being instances of semantic classes, such as the interaction between two proteins (PPIs).

PPI extraction is a complex task since cascades of molecular events are involved which are hard to sort out. Many different approaches have already been tried – pattern-based ones (e.g., by Blaschke et al. (1999), Hakenberg et al. (2005) or Huang et al. (2004)), rule-based ones (e.g., by Yakushiji et al. (2001), Šarić et al. (2004) or Fundel et al. (2007)), and machine learning-based ones (e.g., by Katrenko and Adriaans (2006), Sætre et al. (2007) or Airola et al. (2008)), yet without conclusive results.

In the following, we present our approach to solve Task 1 within the "BioNLP'09 Shared Task on Event Extraction".[1] Task 1 "Event detection and characterization" required to determine the intended relation given *a priori* supplied protein annotations. Our approach considers dependency graphs as the central data structure on which various trimming operations are performed involving syntactic simplification but also, even more important, semantic enrichment by conceptual overlays. A description of the component subtasks is provided in Section 2, while the methodologies intended to solve each subtask are discussed in Section 3. The system pipeline for event extraction reflecting the task decomposition is described in Section 4, while Section 5 provides the evaluation results for our approach.

2 Event Extraction Task

Event extraction is a complex task that can be subdivided into a number of subtasks depending on whether the focus is on the event itself or on the arguments involved:

Event trigger identification deals with the large variety of alternative verbalizations of the same event type, i.e., whether the event is expressed in

[1] http://www-tsujii.is.s.u-tokyo.ac.jp/GENIA/SharedTask/

a verbal or in a nominalized form (e.g., "*A is expressed*" and "*the expression of A*" both refer to the same event type, *viz.* $expression(A)$). Since the same trigger may stand for more than one event type, event trigger ambiguity has to be resolved as well.

Event trigger disambiguation selects the correct event name from the set of alternative event triggers.

Event typing, finally, deals with the semantic classification of a disambiguated event name and the assignment to an event type category.[2]

Argument identification is concerned with finding all necessary participants in an event, i.e., the arguments of the relation.

Argument typing assigns the correct semantic category (entity class) to each of the determined participants in an event (which can be considered as instances of that class).

Argument ordering assigns each identified participant its functional role within the event, mostly Agent (and Patient/Theme).

The sentence "*Regulation of **jun** and **fos gene expression** in human monocytes by the macrophage colony-stimulating factor*", e.g., contains mentions of two *Gene Expression* events with respective THEME arguments "*jun*" and "*fos*", triggered in the text by the literal phrase "*gene expression*".

Task 1 of the "BioNLP'09 Shared Task on Event Extraction" was defined in such a way as to identify a proper relation (event) name and link it with its type, plus one or more associated arguments denoting proteins. To focus on relation extraction only no automatic named entity recognition and interpretation had to be performed (subtask 'argument typing' from above); instead candidate proteins were already pre-tagged. The complexity of Task 1 was raised by the condition that not only proteins were allowed to be arguments but also were events.

3 Event Extraction Solution

Our event extraction approach is summarized in Figure 1 and consists of three major streams – first, the detection of lexicalized event triggers (cf. Section 3.1), second, the trimming of dependency graphs which involves pruning irrelevant and semantically enriching relevant lexical material (cf. Section 3.2),

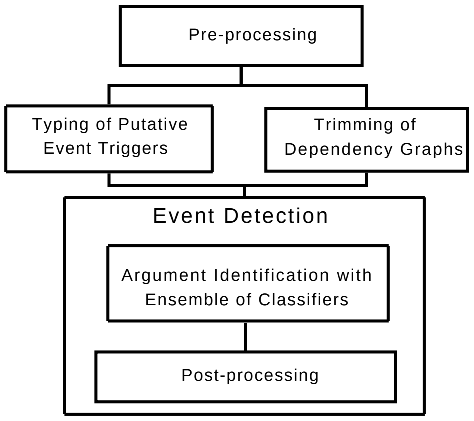

Pre-processing
Typing of Putative Event Triggers
Trimming of Dependency Graphs
Event Detection
Argument Identification with Ensemble of Classifiers
Post-processing

Figure 1: General Architecture of the Event Extraction Solution of the JULIELab Team.

and, third, the identification of arguments for the event under scrutiny (cf. Section 3.3). Event typing results from proper event trigger identification (see Section 3.1.2), which is interlinked with the outcome of the argument identification. We talk about *putative* triggers because we consider, in a greedy manner, all relevant lexical items (see Section 3.1.1) as potential event triggers which might represent an event. Only those event triggers that can eventually be connected to arguments, finally, represent a true event. To achieve this goal we preprocessed both the original training and test data such that we enrich the original training data with automatically predicted event triggers in order to generate more negative examples for a more effective learning of true events.[3]

3.1 Event Trigger Identification

Looking at the wide variety of potential lexicalized triggers for an event, their lacking discriminative power relative to individual event types and their inherent potential for ambiguity,[4] we decided on a dictionary-based approach whose curation principles are described in Section 3.1.1. Our disambiguation policy for the ambiguous lexicalized event trig-

[2]In our approach, event trigger disambiguation already implies event typing.

[3]Although the training data contains cross-sentence event descriptions, our approach to event extraction is restricted to the sentence level only.

[4]Most of the triggers are neither specific for molecular event descriptions, in general, nor for a special event type. "Induction", e.g., occurs 417 times in the training data. In 162 of these cases it acts as a trigger for *Positive_regulation*, 6 times as a trigger for *Transcription*, 8 instances trigger *Gene_expression*, while 241 occurrences do not trigger an event at all.

gers assembled in this suite of dictionaries, one per event type, is discussed in Section 3.1.2.

3.1.1 Manual Curation of the Dictionaries

We started collecting our dictionaries from the original GENIA event corpus (Kim et al., 2008a). The extracted event triggers were then automatically lemmatized[5] and the resulting lemmata were subsequently ranked by two students of biology according to their predictive power to act as a trigger for a particular event type. This expert assessment led us to four trigger groups (for each event type these groups were determined separately):

(1) Triggers are *important* and *discriminative* for a specific event type. This group contains event triggers such as "upregulate" for *Positive_regulation*.

(2) Triggers are *important* though *not fully discriminative* for a particular event type; yet, this deficiency can be overcome by other lexical cues within the context of the same sentence. This group with in-context disambiguators contains lexical items such as "proteolyse" for *Protein_catabolism*.

(3) Triggers are *non-discriminative* for an event type and even cannot be disambiguated by linguistic cues within the context of the same sentence. This group contains lexical items such as "presence" for *Localization* and *Gene_expression*.

(4) Triggers are absolutely *non-discriminative* for an event. This group holds general lexical triggers such as "observe", "demonstrate" or "function".

The final dictionaries used for the detection of putative event triggers are a union of the first two groups. They were further extended by biologists with additional lexical material of the first group. The dictionaries thus became event type-specific – they contain all morphological forms of the original lemma, which were automatically generated using the Specialist NLP Tools (2008 release).

We matched the entries from the final set of dictionaries with the shared task data using the Lingpipe Dictionary Chunker.[6] After the matching process, some cleansing had to be done.[7]

3.1.2 Event Trigger Disambiguation

Preliminary experiments indicated that the disambiguation of event triggers might be beneficial for the overall event extraction results since events tend to be expressed via highly ambiguous triggers. Therefore, we performed a disambiguation step preceding the extraction of any argument structures.

It is based on the *importance* of an event trigger t_i for a particular event type T as defined by $Imp(t_i^T) := \frac{f(t_i^T)}{\sum_i f(t_i^T)}$, where $f(t_i^T)$ is the frequency of the event trigger t_i of the selected event type T in a training corpus divided by the total amount of all event triggers of the selected event type T in that training corpus. The frequencies are measured on stemmed event triggers. For example, Imp for the trigger stem "*depend*" amounts to 0.013 for the event type *Positive_regulation*, while for the event type *Regulation* it yields 0.036 . If a text span contains several event triggers with the same span offset, the event trigger with $max(Imp)$ is selected and other putative triggers are discarded. The trigger stem "*depend*" remains thus only for *Regulation*.

3.2 Trimming Dependency Graphs

When we consider event (relation) extraction as a semantic interpretation task, plain dependency graphs as they result from deep syntactic parsing might not be appropriate to directly extract semantic information from. This is due to two reasons - they contain a lot of apparently irrelevant lexical nodes (from the semantic perspective of event extraction) and they also contain much too specific lexical nodes that might better be grouped and further enriched semantically. Trimming dependency graphs for the purposes of event extraction, therefore, amounts to eliminate semantically irrelevant and to semantically enrich relevant lexical nodes (i.e., overlay with concepts). This way, we influence the final representation for the machine learners we employ (in terms of features or kernel-based representations) — we may avoid an overfitting of the feature or kernel spaces with syntactic and lexical data and thus reduce structural information in a linguistically motivated way.

[5]We used the lemmatizer from the Specialist NLP Tools (`http://lexsrv3.nlm.nih.gov/SPECIALIST/index.html`, 2008 release).

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

[7]Event triggers were removed which (1) were found within sentences without any protein annotations, (2) occurred within a longer event trigger, (3) overlapped with a longer trigger of the same event type, (4) occurred inside an entity mention annotation.

3.2.1 Syntactic Pruning

Pruning targets auxiliary and modal verbs which govern the main verb in syntactic structures such as passives, past or future tense. We delete the auxiliars/modals as govenors of the main verbs from the dependency graph and propagate the semantics-preserving dependency relations of these nodes directly to the main verbs. Adhering to the dependency tree format and labeling conventions set up for the 2006 and 2007 CoNLL shared tasks on dependency parsing main verbs are usually connected with the auxiliar by the VC dependency relation (see Figure 2). Accordingly, in our example, the verb "*activate*" is promoted to the ROOT in the dependency graph and governs all nodes that were originally governed by the modal "*may*".

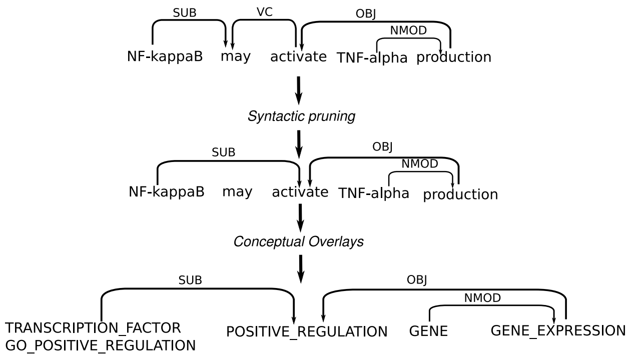


Figure 2: Trimming of Dependency Graphs.

3.2.2 Conceptual Decoration

Lexical nodes in the (possibly pruned) dependency graphs deemed to be important for argument extraction were then enriched with semantic class annotations, instead of keeping the original lexical (stem) representation (see Figure 2). The rationale behind this decision was to generate more powerful kernel-based or features representations (see Section 3.3.2 and 3.3.1).

The whole process is based on a three-tier task-specific semantic hierarchy of named entity classes. The top rank is constituted by the equivalent classes *Transcription factor*, *Binding site*, and *Promoter*. The second rank is occupied by MESH terms, and the third tier assembles the named entity classes *Gene* and *Protein*. Whenever a lexical item is categorized by one of these categories, the associated node in the dependency graph is overlaid with that category applying the ranking in cases of conflicts.

We also enriched the gene name mentions with their respective Gene Ontology Annotations from GOA.[8] For this purpose, we first categorized GO terms both from the "molecular function" and from the "biological process" branch with respect to their matching event type, e.g., *Phosphorylation* or *Positive_regulation*. We then mapped all gene name mentions which occurred in the text to their UNIPROT identifier using the gene name normalizer GENO (Wermter et al., 2009). This identifier links a gene with a set of (curated) GO annotations.

In addition, we inserted semantic information in terms of the event trigger type and the experimental methods. As far as experimental methods are concerned, we extracted all instances of them annotated in the GENIA event corpus. One student of biology sorted the experimental methods relative to the event categories under scrutiny. For example "*affinity chromatography*" was assigned both to the *Gene_expression* and to the *Binding* category. For our purposes, we only included those GO annotations and experimental methods which matched the event types to be identified in a sentence.

3.3 Argument Identification and Ordering

The argument identification task can be subdivided into three complexity levels. Level (1) incorporates five event types (*Gene_expression*, *Transcription*, *Protein_catabolism*, *Localization*, *Phosphorylation*) which involve a single participant with a THEME role only. Level (2) is concerned with one event type (*Binding*) that provides an n-ary argument structure where all arguments occupy the THEME(*n*) role. Level (3) comprises three event types (*Positive_regulation*, *Negative_regulation*, or an unspecified *Regulation*) that represent a regulatory relation between the above-mentioned event classes or proteins. These events have usually a binary structure, with a THEME argument and a CAUSE argument.

For argument extraction, we built sentence-wise pairs of putative triggers and their putative argument(s), the latter involving ontological information about the event type. For Level (1), we built pairs only with proteins, while for Level (3) we al-

[8]http://www.ebi.ac.uk/GOA

lowed all events as possible arguments. For Level (2), *Binding* events, we generated binary (trigger, protein) pairs as well as triples (trigger, protein_1, protein_2) to adequately represent the binding between two proteins.[9] Pairs of mentions not connected by a dependency path could not be detected. For the argument extraction we chose two machine learning-based approaches, feature-based and a kernel-based one, as described below.[10]

3.3.1 Feature-based Classifier

We distinguished three groups of features. First, *lexical* features (covering lexical items before, after and between both mentions (of the event trigger and an argument) as described by Zhou and Zhang (2007)); second, *chunking* features (concerned with head words of the phrases between two mentions as described by Zhou and Zhang (2007)); third, *dependency parse* features (considering both the selected dependency levels of the arguments (parents and least common subsumer) as discussed by Katrenko and Adriaans (2006), as well as a shortest dependency path structure between the arguments as used by Kim et al. (2008b) for *walk* features).

For the feature-based approach, we chose the Maximum Entropy (ME) classifier from MALLET.[11]

3.3.2 Graph Kernel Classifier

The graph kernel uses a converted form of dependency graphs in which each dependency node is represented by a set of labels associated with that node. The dependency edges are also represented as nodes in the new graph such that they are connected to the nodes adjacent in the dependency graph. Subgraphs which represent, e.g., the linear order of the words in the sentence can be added, if required. The entire graph is represented in terms of an adjacency matrix which is further processed to contain the summed weights of paths connecting two nodes of the graph (see Airola et al. (2008) for details).

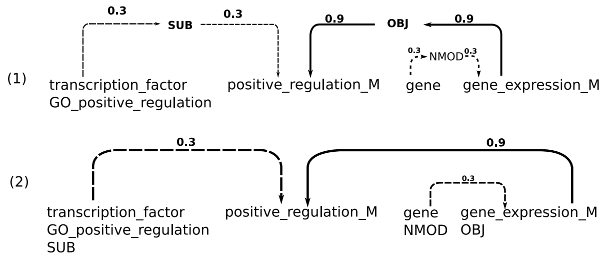


Figure 3: Graph Kernel Representation for a Trimmed Dependency Graph — (1) original representation, (2) representation without graph dependency edge nodes (weights (0.9, 0.3) taken from Airola et al. (2008)).

For our experiments, we tried some variants of the original graph kernel. In the original version each dependency graph edge is represented as a node. That means that connections between graph token nodes are expressed through *graph dependency edge nodes* (see Figure 3; (1)). To represent the connections between original tokens as direct connections in the graph, we removed the edge nodes and each token was assigned the edge label (its dependency label; see Figure 3; (2)). Further variants included encodings for (1) the shortest dependency path (*sp*) between two mentions (argument and trigger)[12] (2) the complete dependency graph (*sp-dep*), and (3) the complete dependency graph and linear information (*sp-dep-lin*) (the original configuration from Airola et al. (2008)).

For the graph kernel, we chose the LibSVM (Chang and Lin, 2001) Support Vector Machine as classifier.

3.4 Postprocessing

The postprocessing step varies for the three different Levels (see Section 3.3). For every event trigger of Level (1) (e.g., *Gene_expression*), we generate one event per relation comprising a trigger and its argument. For Level (2) (*Binding*), we create a *Binding* event with two arguments only for triples (trigger, protein_1, protein_2). For the third Level, we create for each event trigger and its associated arguments $e = n \times m$ events, for n CAUSE arguments and m THEME arguments.

[9]We did not account for the binding of more than two proteins as this would have led to a combinatory explosion of possible classifications.

[10]In our experiments, we used full conceptual overlaying (see Section 3.2) for the kernel-based representation and partial overlaying for the dependency parse features (only gene/protein annotation was exploited here). Graph representations allow for many semantic labels to be associated with a node.

[11]`http://mallet.cs.umass.edu/index.php/Main_Page`

[12]For *Binding* we extracted the shortest path between two protein mentions if we encounter a triple (trigger, protein_1, protein_2).

4 Pipeline

The event extraction pipeline consists of two major parts, a pre-processor and the dedicated event extractor. As far as pre-processing is concerned, we imported the sentence splitting, tokenization and GDep parsing results (Sagae and Tsujii, 2007) as prepared by the shared task organizers for all data sets (training, development and test). We processed this data with the OpenNLP POS tagger and Chunker, both re-trained on the GENIA corpus (Buyko et al., 2006). Additionally, we enhanced the original tokenization by one which includes hyphenization of lexical items such as in "*PMA-dependent*". [13]

The data was further processed with the gene normalizer GENO(Wermter et al., 2009) and a number of regex- and dictionary-based entity taggers (covering promoters, binding sites, and transcription factors). We also enriched gene name mentions with their respective Gene Ontology annotations (see Section 3.2.2). The MESH thesaurus (except chemical and drugs branch) was mapped on the data using the Lingpipe Dictionary Chunker.[14]

After preprocessing, event extraction was started distinguishing between the event trigger recognition (cf. Section 3.1), the trimming of the dependency graphs (cf. Section 3.2), and the argument extraction proper (cf. Section 3.3).[15] We determined in our experiments on the development data the performance of every classifier type and its variants (for the graph kernel), and of ensembles of the most performant (F-Score) graph kernel variant and an ME model.[16] We present here the argument extraction configuration used for the official run.[17] For the prediction of *Phosphorylation*, *Localization*, *Protein_catabolism* types we used the graph kernel in its "*sp without dependency-edge-nodes*" configuration, while for the prediction of *Transcription* and *Gene_expression* events we used an ensemble of the graph kernel in its "*sp with dependency-edge-nodes*" variant, and an ME model. For the prediction of *Binding* we used an ensemble of the graph kernel ("*sp-dep with dependency-edge-nodes*") and an ME model. For the prediction of regulatory events we used ME models for each regulatory type.

[13]This tokenization is more advantageous for the detection of additional event triggers as it allows to generate dependency relations from hyphenated terms. For example, in "*PMA-dependent*", "*PMA*" will be a child of "*dependent*" linked by the AMOD dependency relation, and "*dependent*" receives the original dependency relation of the "*PMA-dependent*" token.

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

[15]For the final configurations of the graph kernel, we optimized the C parameter in the spectrum between 2^{-3} and 2^3 on the final training data for every event type separately.

[16]In the *ensemble* configuration we built the union of positive instances.

[17]We achieved with this configuration the best performance on the development set.

5 Results

The baseline against which we compared our approach can be captured in a single rule. We extract for every pair of a putative trigger and a putative argument the shortest dependency path between them. If the shortest dependency path does not contain any direction change, i.e., the argument is either a direct child or a direct parent of the trigger, and if the path does not contain any other intervening event triggers, the argument is taken as the THEME role.

We performed evaluations on the shared task development and test set. Our baseline achieved competitive results of 36.0% precision, 34.0% recall, 35.0% F-score on the development set (see Table 1), and 30.4% precision, 35.7% recall, 32,8% F-score on the test set (see Table 2). In particular the one-argument events, i.e., *Gene_expression*, *Protein_catabolism*, *Phosphorylation* are effectively extracted with an F-score around 70.0%. More complex events, in particular events of Level (3), i.e., (*Regulation*) were less properly dealt with because of their strong internal complexity.

Event Class	**gold**	**recall**	**prec.**	**F-score**
Localization	53	75.47	30.30	43.24
Binding	248	33.47	20.80	25.66
Gene_expression	356	76.12	75.07	75.59
Transcription	82	68.29	40.58	50.91
Protein_catabolism	21	76.19	66.67	71.11
Phosphorylation	47	76.60	72.00	74.23
Regulation	169	14.20	15.09	14.63
Positive_regulation	617	15.40	20.83	17.71
Negative_regulation	196	11.73	13.22	12.43
TOTAL	1789	36.00	34.02	34.98

Table 1: Baseline results on the shared task development data. Approximate Span Matching/Approximate Recursive Matching.

Event Class	gold	recall	prec.	F-score	gold	recall	prec.	F-score
Localization	174	42.53	44.85	43.66	174	42.53	44.85	43.66
Binding	347	32.28	37.09	34.51	398	44.22	58.28	50.29
Gene_expression	722	61.36	80.55	69.65	722	61.36	80.55	69.65
Transcription	137	39.42	35.06	37.11	137	39.42	35.06	37.11
Protein_catabolism	14	71.43	66.67	68.97	14	71.43	66.67	68.97
Phosphorylation	135	65.93	90.82	76.39	135	65.93	90.82	76.39
EVT-TOTAL	1529	51.14	60.90	55.60	1580	53.54	65.89	59.08
Regulation	291	9.62	11.72	10.57	338	9.17	12.97	10.75
Positive_regulation	983	10.38	11.33	10.83	1186	14.67	19.33	16.68
Negative_regulation	379	14.25	19.22	16.36	416	14.18	21.00	16.93
REG-TOTAL	1653	11.13	12.96	11.98	1940	13.61	18.59	15.71
ALL-TOTAL	3182	30.36	35.72	32.82	3520	31.53	41.05	35.67

Table 2: Baseline results on the shared task test data. Approximate Span Matching/Approximate Recursive Matching (columns 3-5). Event decomposition, Approximate Span Matching/Approximate Recursive Matching (columns 7-9).

The event extraction approach, in its final configuration (see Section 4), achieved a performance of 50.4% recall, 45.8% precision and 48.0% F-score on the development set (see Table 4), and 45.8% recall, 47.5% precision and 46.7% F-score on the test set (see Table 3). This approach clearly outperformed the baseline with an increase of 14 percentage points on the test data. In particular, the events of Level (2) and (3) were more properly dealt with than by the baseline. In the event decomposition mode (argument detection is evaluated in a decomposed event) we achieved a performance of 49.4% recall, 56.2% precision, and 52.6% F-score (see Table 3).

Our experiments on the development set showed that the combination of the feature-based and the graph kernel-based approach can boost the results up to 6 percentage points F-score (for the *Binding* event type). It is interesting that the combination for *Binding* increased recall without dropping precision. The original graph kernel approach for *Binding* events performs with 38.3% recall, 27.9% precision and 32.3% F-score on the development set. The combined approach comes with a remarkable increase of 14 percentage points in recall. The combination could also boost the recall of the *Gene_expression* and *Transcription* by 15 percentage points and 5 percentage points, respectively, without seriously dropping the precision (4 points for every type). For the other event types, no improvements were found when we combined both approaches.

5.1 Error Discussion

One expert biologist analyzed 30 abstracts randomly extracted from the development error data. We determined seven groups of errrors based on this analysis. The first group contains examples for which an event should be determined, but a false argument was found (e.g., *Binding* arguments were not properly sorted, or correct and false arguments were detected for the same trigger) (44 examples). The second group comprised examples where no trigger was found (23 examples). Group (3) stands for cases where no events were detected although a trigger was properly identified (14 examples). Group (4) holds examples detected in sentences which did not contain any events (12 examples). Group (5) lists biologically meaningful analyses, actually very close to the gold annotation, especially for the cascaded regulatory events (12 examples), while Group (6) incorporates examples of a detected event with incorrect type (1 example). Group (7) gathers misleading gold annotations (10 examples).

This assessment clearly indicates that a major source of errors can be traced to the level of argument identification, in particular for *Binding* events. The second major source has its offspring at the level of trigger detection (we ignored, for example, triggers such as "*in the presence of*", "*when*", "*normal*"). About 10% of the errors are due to a slight difference between extracted events and gold events. For example, in the phrase "*role for NF-kappaB in the regulation of FasL expression*" we

Event Class	**gold**	**recall**	**prec.**	**F-score**	**gold**	**recall**	**prec.**	**F-score**
Localization	174	43.68	77.55	55.88	174	43.68	77.55	55.88
Binding	347	49.57	35.25	41.20	398	63.57	54.88	58.91
Gene_expression	722	64.82	80.27	71.72	722	64.82	80.27	71.72
Transcription	137	35.77	62.03	45.37	137	35.77	62.03	45.37
Protein_catabolism	14	78.57	84.62	81.48	14	78.57	84.62	81.48
Phosphorylation	135	76.30	91.15	83.06	135	76.30	91.15	83.06
EVT-TOTAL	1529	57.49	63.97	60.56	1580	60.76	71.27	65.60
Regulation	291	31.27	30.13	30.69	338	35.21	37.54	36.34
Positive_regulation	983	34.08	37.18	35.56	1186	40.64	49.33	44.57
Negative_regulation	379	40.37	31.16	35.17	416	42.31	39.11	40.65
REG-TOTAL	1653	35.03	34.18	34.60	1940	40.05	44.55	42.18
ALL-TOTAL	**3182**	**45.82**	**47.52**	**46.66**	**3520**	**49.35**	**56.20**	**52.55**

Table 3: Offical Event Extraction results on the shared task test data of the JULIELab Team. Approximate Span Matching/Approximate Recursive Matching (columns 3-5). Event decomposition, Approximate Span Matching/Approximate Recursive Matching (columns 7-9).

could not extract the gold event *Regulation* of *Regulation (Gene_expression (FasL))* associated with the trigger "*role*", but we were able to find the (inside) event *Regulation (Gene_expression (FasL))* associated with the trigger "*regulation*". Interestingly, the typing of events is not an error source in spite of the simple disambiguation approach. Still, our disambiguation strategy is not appropriate for the analysis of *double-annotated* triggers such as "overexpression", "transfection", etc., which are annotated as *Gene_expression* and *Positive_regulation* and are a major source of errors in Group (2). As Group (6) is an insignificant source of errors in our randomly selected data, we focused our error analysis on the especially ambiguous event type *Transcription*. We found from 34 errors that 14 of them were due to the disambiguation strategy (in particular for triggers "(gene) expression" and "induction").

Event Class	**gold**	**recall**	**prec.**	**F-score**
Localization	53	71.70	74.51	73.08
Binding	248	52.42	29.08	37.41
Gene_expression	356	75.28	81.46	78.25
Transcription	82	60.98	73.53	66.67
Protein_catabolism	21	90.48	79.17	84.44
Phosphorylation	47	82.98	84.78	83.87
Regulation	169	37.87	36.78	37.32
Positive_regulation	617	34.36	35.99	35.16
Negative_regulation	196	41.33	33.61	37.07
TOTAL	**1789**	**50.36**	**45.76**	**47.95**

Table 4: Event extraction results on the shared task development data of the official run of the JULIELab Team. Approximate Span Matching/Approximate Recursive Matching.

6 Conclusion

Our approach to event extraction incorporates manually curated dictionaries and machine learning methodologies to sort out associated event triggers and arguments on trimmed dependency graph structures. Trimming combines pruning irrelevant lexical material from a dependency graph and decorating particularly relevant lexical material from that graph with more abstract conceptual class information. Given that methodological framework, the JULIELab Team scored on 2nd rank among 24 competing teams, with 45.8% precision, 47.5% recall and 46.7% F1-score on all 3,182 events.

7 Acknowledgments

We wish to thank Rico Landefeld for his technical support, Tobias Wagner and Rico Pusch for their constant help and great expertise in biological issues. This research was partially funded within the BOOTSTREP project under grant FP6-028099 and the CALBC project under grant FP7-231727.

References

Antti Airola, Sampo Pyysalo, Jari Björne, Tapio Pahikkala, Filip Ginter, and Tapio Salakoski. 2008. A graph kernel for protein-protein interaction extraction. In *Proceedings of the Workshop on Current Trends in Biomedical Natural Language Processing*, pages 1–9.

Christian Blaschke, Miguel A. Andrade, Christos Ouzounis, and Alfonso Valencia. 1999. Automatic extraction of biological information from scientific text: Protein-protein interactions. In *ISMB'99 – Proceedings of the 7th International Conference on Intelligent Systems for Molecular Biology*, pages 60–67.

Ekaterina Buyko, Joachim Wermter, Michael Poprat, and Udo Hahn. 2006. Automatically adapting an NLP core engine to the biology domain. In *Proceedings of the Joint BioLINK-Bio-Ontologies Meeting. A Joint Meeting of the ISMB Special Interest Group on Bio-Ontologies and the BioLINK Special Interest Group on Text Data M ining in Association with ISMB*, pages 65–68. Fortaleza, Brazil, August 5, 2006.

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`.

Katrin Fundel, Robert Küffner, and Ralf Zimmer. 2007. Relex-relation extraction using dependency parse trees. *Bioinformatics*, 23(3):365–371.

Jörg Hakenberg, Ulf Leser, Conrad Plake, Harald Kirsch, and Dietrich Rebholz-Schuhmann. 2005. LLL'05 challenge: Genic interaction extraction - identification of language patterns based on alignment and finite state automata. In *Proceedings of the 4th Learning Language in Logic Workshop (LLL05)*, pages 38–45.

Minlie Huang, Xiaoyan Zhu, Donald G. Payan, Kunbin Qu, and Ming Li. 2004. Discovering patterns to extract protein-protein interactions from full texts. *Bioinformatics*, 20(18):3604–3612.

Sophia Katrenko and Pieter W. Adriaans. 2006. Learning relations from biomedical corpora using dependency trees. In Karl Tuyls, Ronald L. Westra, Yvan Saeys, and Ann Nowé, editors, *KDECB 2006 – Knowledge Discovery and Emergent Complexity in Bioinformatics. Revised Selected Papers of the 1st International Workshop.*, volume 4366 of *Lecture Notes in Computer Science*, pages 61–80. Ghent, Belgium, May 10, 2006. Berlin: Springer.

Jin-Dong Kim, Tomoko Ohta, and Jun'ichi Tsujii. 2008a. Corpus annotation for mining biomedical events from literature. *BMC Bioinformatics*, 9(10).

Seon-Ho Kim, Juntae Yoon, and Jihoon Yang. 2008b. Kernel approaches for genic interaction extraction. *Bioinformatics*, 24(1):118–126.

Rune Sætre, Kenji Sagae, and Jun'ichi Tsujii. 2007. Syntactic features for protein-protein interaction extraction. In Christopher J. O. Baker and Jian Su, editors, *LBM 2007*, volume 319, pages 6.1–6.14.

Kenji Sagae and Jun'ichi Tsujii. 2007. Dependency parsing and domain adaptation with LR models and par ser ensembles. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 1044–1050.

Jasmin Šarić, Lars J. Jensen, Rossitza Ouzounova, Isabel Rojas, and Peer Bork. 2004. Extracting regulatory gene expression networks from pubmed. In *ACL '04: Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, page 191, Morristown, NJ, USA. Association for Computational Linguistics.

Joachim Wermter, Katrin Tomanek, and Udo Hahn. 2009. High-performance gene name normalization with GeNo. *Bioinformatics*, 25(6):815–821.

Akane Yakushiji, Yuka Tateisi, Yusuke Miyao, and Jun'ichi Tsujii. 2001. Event extraction from biomedical papers using a full parser. In Russ B. Altman, A. Keith Dunker, Lawrence Hunter, Kevin Lauderdale, and Teri E. Klein, editors, *PSB 2001 – Proceedings of the 6th Pacific Symposium on Biocomputing*, pages 408–419. Maui, Hawaii, USA. January 3-7, 2001. Singapore: World Scientific Publishing.

Guodong Zhou and Min Zhang. 2007. Extracting relation information from text documents by exploring various types of knowledge. *Information Processing & Management*, 43(4):969–982.

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/

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/`

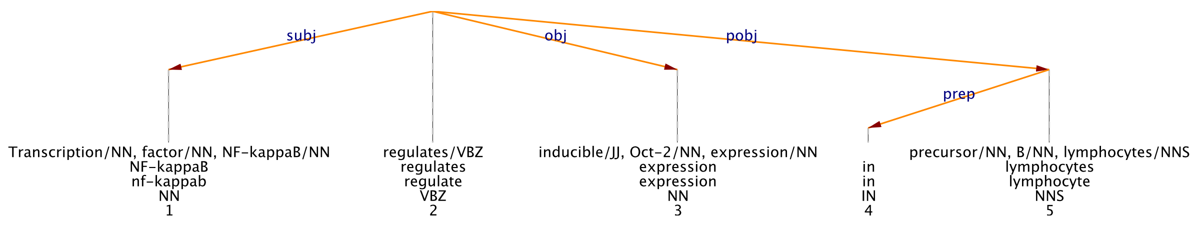


Figure 1: Dependency-syntax tree of the title of abstract 9360945: "Transcription factor NF-kappaB regulates inducible Oct-2 gene expression in precursor B lymphocytes." The dependency relations link together the heads of the 5 chunks.

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:

$$p(Trigger \mid Token) = \frac{f(Token \wedge (Token = Trigger))}{f(Token)} \quad (1)$$

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.

$$p(EventStruct \mid Trigger) = \frac{f(Trigger \wedge EventStruct)}{f(Trigger)} \quad (2)$$

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

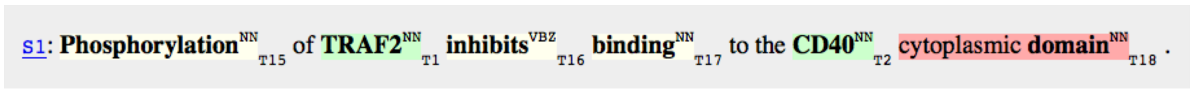

s1: Phosphorylation$^{NN}_{T15}$ of TRAF2$^{NN}_{T1}$ inhibits$^{VBZ}_{T16}$ binding$^{NN}_{T17}$ to the CD40$^{NN}_{T2}$ cytoplasmic domain$^{NN}_{T18}$.

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.

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

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.

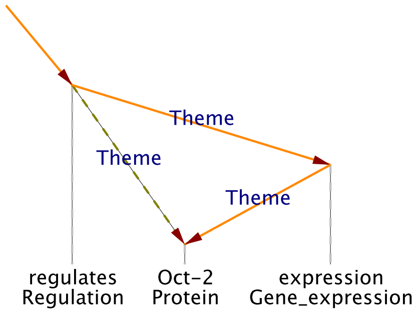


Figure 3: Visualization of two simple event structures *regulates(Oct-2)* and *expression(Oct-2)*, and a complex structure *regulates(expression(Oct-2))*.

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 post-processing 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.

Biomedical Event Extraction without Training Data

Andreas Vlachos, Paula Buttery, Diarmuid Ó Séaghdha, Ted Briscoe
Computer Laboratory
University of Cambridge
Cambridge, UK
av308,pjb48,do242,ejb@cl.cam.ac.uk

Abstract

We describe our system for the BioNLP 2009 event detection task. It is designed to be as domain-independent and unsupervised as possible. Nevertheless, the precisions achieved for single theme event classes range from 75% to 92%, while maintaining reasonable recall. The overall F-scores achieved were 36.44% and 30.80% on the development and the test sets respectively.

1 Introduction

In this paper we describe the system built for the BioNLP 2009 event detection and characterization task (Task 1). The approach is based on the output of a syntactic parser and standard linguistic processing, augmented by rules acquired from the development data. The key idea is that a trigger connected with an appropriate argument along a path through the syntactic dependency graph forms an event.

The goal we set for our approach was to avoid using training data explicitly annotated for the task and to preserve domain independence. While we acknowledge the utility of supervision (in the form of annotated data) and domain knowledge, we believe it is valuable to explore an unsupervised approach. Firstly, manually annotated data is expensive to create and the annotation process itself is difficult and unavoidably results in inconsistencies, even in well-explored tasks such as named entity recognition (NER). Secondly, unsupervised approaches, even if they fail to reach the performance of supervised ones, are likely to be informative in identifying useful features for the latter. Thirdly, exploring the potential of such a system may highlight what domain knowledge is useful and its potential contribution to performance. Finally, preserving domain independence allows us to develop and evaluate a system that could be used for similar tasks with minimal adaptation.

The overall architecture of the system is as follows. Initiallly, event triggers are identified and labelled with event types using seed terms. Based on the dependency output of the parser the triggers are connected with candidate arguments using patterns identified in the development data. Anaphoric candidate arguments are then resolved. Finally, the triggers connected with appropriate arguments are post-processed to generate the final set of events. Each of these stages are described in detail in subsequent sections, followed by experiments and discussion.

2 Trigger identification

We perform trigger identification using the assumption that events are triggered in text either by verbal or nominal prdicates (Cohen et al., 2008).

To build a dictionary of verbs and their associated event classes we use the triggers annotated in the training data. We lemmatize and stem the triggers with the morphology component of the RASP toolkit (Briscoe et al., 2006)[1] and the Porter stemmer[2] respectively. We sort the trigger stem - event class pairs found according to their frequency in the training data and we keep only those pairs that appear at least 10 times. The trigger stems are then mapped to verbs. This excludes some relatively common triggers, which will reduce recall, but, given that we rely exclusively on the parser for

[1]http://www.cogs.susx.ac.uk/lab/nlp/rasp/

[2]http://www.tartarus.org/˜martin/PorterStemmer

argument extraction, such triggers would be difficult to handle. For verbs with more than one event class we keep only the most frequent one.

We consider the assumption that each verb denotes a single event class to be a reasonable one given the restricted task domain. It hinders us from dealing with triggers denoting multiple event classes but it simplifies the task so that we do not need annotated data. While we use the training data triggers to obtain the list of verbs and their corresponding event types, we believe that such lists could be obtained by clustering (Korhonen et al., 2008) with editing and labelling by domain experts. This is the only use of the training data we make in our system.

During testing, using the tokenized text provided, we attempt to match each token with one of the verbs associated with an event type. We perform this by relaxing the matching successively, using the token lemma, then stem, and finally allowing a partial match in order to deal with particles (so that e.g. *co-transfect* matches *transfect*). This process returns single-token candidate triggers which, while they do not reproduce the trigger annotation, are likely to be adequate for event extraction. We overgenerate triggers, since not all occurrences denote an event, either because they are not connected with appropriate arguments or because they are found in a non-event denoting context, but we expect to filter these at the argument extraction stage.

3 Argument extraction

Given a set of candidate triggers, we attempt to connect them with appropriate arguments using the dependency graph provided by a parser. In our experiments we use the domain-independent unlexicalized RASP parser, which generates parses over the part-of-speech (PoS) tags of the tokens generated by an HMM-based tagger trained on balanced English text. While we expect that a parser adapted to the biomedical domain may perform better, we want to preserve the domain-independence of the system and explore its potential.

The only adjustment we make is to change the PoS tags of tokens that are part of a protein name to proper names tags. We consider such an adjustment domain-independent given that NER is available in many domains (Lewin, 2007). Following Haghighi et al (2005), in order to ameliorate parsing errors, we use the top-10 parses and return a set of bilexical head-dependent grammatical relations (GRs) weighted according to the proportion and probability of the top parses supporting that GR.

The GRs produced by the parser define directed graphs between tokens in the sentence, and a partial event is formed when a path that connects a trigger with an appropriate argument is identified. GR paths that are likely to generate events are selected using the development data, which does not contradict the goals of our approach because we do not require annotated training data. Development data is always needed in order to build and test a system, and such supervision could be provided by a human expert, albeit not as easily as for the list of trigger verbs. The set of GR paths identified follow:

VERB-TRIGGER –subject– ARG
NOUN-TRIGGER –iobj– PREP –dobj– ARG
NOUN-TRIGGER –modifier– ARG
TRIGGER –modifier– PREP –obj– ARG
TRIGGER –passive subject– ARG

The final system uses three sets of GR paths: one for Regulation events; one for Binding events; and one for all other events. The difference between these sets is in the lexicalization of the linking prepositions. For example, in Binding events the linking preposition required lexicalization since *binds x to/with y* denotes a correct event but not *binds x by y*. Binding events also required additional GR paths to capture constructions such as *binding of x to y*. For Regulation events, the path set was further augmented to differentiate between theme and cause. When the lexicalized GR pattern sets yielded no events we backed-off to the unlexicalized pattern set, which is identical for all event types. In all GR path sets, the trigger was unlexicalized and only restricted by PoS tag.

4 Anaphora resolution

The events and arguments identified in the parsed abstracts are post-processed in context to identify protein referents for event arguments that are anaphoric (e.g., *these proteins*, *its phosphorylation*) or too complex to be extracted directly from the grammatical relations (*phosphorylation of cellular proteins , notably phospholipase C gamma 1*). The anaphoric linking is performed by a set of heuristic rules manually designed to capture a number of common cases observed in the development dataset. A further phenomenon dealt with by rules is coreference between events, for example in *The expression of LAL-mRNA is induced. This induction is dependent on…* where the Induction event described by the first sentence is the same as the theme of the Regulation event in the second and should be given the same event index. The development of the post-processing rules favoured precision over recall, but the low frequency of each case considered means that some overfitting to the development data may have been unavoidable.

5 Event post-processing

At the event post-processing stage, we form complete events considering the trigger-argument pairs produced at the argument extraction stage whose arguments are resolved (possibly using anaphora resolution) either to a protein name or to a candidate trigger. The latter are considered only for regulation event triggers. Furthermore, regulation event trigger-argument pairs are tagged either as theme or cause at the argument extraction stage.

For each non-regulation trigger-argument pair, we generate a single event with the argument marked as theme. Given that we are dealing only with Task 1, this approach is expected to deal adequately with all event types except Binding, which can have multiple themes. Regulation events are formed in the following way. Given that the cause argument is optional, we generate regulation events for trigger-argument pairs whose argument is a protein name or a trigger that has a formed event. Since regulation events can have other regulation events as themes, we repeat this process until no more events can be formed. Occasionally, the use of multiple parses results in cycles between regulation triggers which are resolved using the weighted GR scores. Then, we attach any cause arguments that share the same trigger with a formed regulation event.

In the analysis performed for trigger identification in Section 2, we observed that certain verbs were consistently annotated with two events (namely *overexpress* and *transfect*), a non-regulation event and a regulation event with the former event as its theme. For candidate triggers that were recognized due to such verbs, we treat them as non-regulation events until the post-processing stage where we generate two events.

6 Experiments - Discussion

We expected that our approach would achieve high precision but relatively low recall. The evaluation of our final submissions on the development and test data (Table 1) confirmed this to a large extent. For the non-regulation event classes excluding Binding, the precisions achieved range from 75% to 92% in both development and test data, with the exception of Transcription in the test data. Our approach extracts Binding events with a single theme, more suitably evaluated by the Event Decomposition evaluation mode in which a similar high precision/low recall trend is observed, albeit with lower scores.

Of particular interest are the event classes for which a single trigger verb was identified, namely Transcription, Protein catabolism and Phosphorylation, which makes it easier to identify the strengths and weaknesses of our approach. For the Phosphorylation class, almost all the triggers that were annotated in the training data can be captured using the verb *phosporylate* and as a result, the performances achieved by our system are 70.59% and 60.63% F-score on the development and test data respectively. The precision was approximately 78% in both datasets, while recall was lower due to parser errors and unresolved anaphoric references. For the Protein catabolism class, *degrade* was identified as the only trigger verb, resulting in similar high precision but relatively lower recall due to the higher lexical variation of the triggers for this class. For the Transcription class we considered only *transcribe* as a trigger verb, but while the performance on the development data is reasonable (55%), the performance on the test data is substantially lower (20%). Inspecting the event triggers in the training data reveals that some very common triggers for this class either cannot be mapped to a verb (e.g., *mrna*) or are commonly used as triggers for other event classes. A notable case of the latter type is the verb *express*, which, while mostly a Gene Expressions trigger, is also annotated as Transcription more than 100 times in the training data. Assuming that this is desirable,

	Development			Test		
Event Class	recall	precision	fscore	recall	precision	fscore
Localization	45.28	92.31	60.76	25.86	90.00	40.18
Binding	12.50	24.41	16.53	12.68	31.88	18.14
Gene expression	52.25	80.79	63.46	45.57	75.81	56.92
Transcription	42.68	77.78	55.12	12.41	56.67	20.36
Protein catabolism	42.86	81.82	56.25	35.71	83.33	50.00
Phosphorylation	63.83	78.95	70.59	49.63	77.91	60.63
Event Total	39.03	65.97	49.05	33.16	68.15	44.61
Regulation	20.12	50.75	28.81	9.28	36.49	14.79
Positive regulation	16.86	48.83	25.06	11.39	38.49	17.58
Negative regulation	11.22	36.67	17.19	6.86	36.11	11.53
Regulation Total	16.29	47.06	24.21	9.98	37.76	15.79
Total	26.55	58.09	36.44	21.12	56.90	30.80
Binding (decomposed)	26.92	66.14	38.27	18.84	54.35	27.99

Table 1: Performance analysis on development and test data using Approximate Span/Partial Recursive Matching.

a more appropriate solution would need to take context into account.

Our performance on the regulation events is substantially lower in both recall and precision. This is expected, as they rely on the extraction of non-regulation events. The variety of lexical triggers is not causing the drop in performance though, since our system performed reasonably well in the Gene Expression and Localization classes which have similar lexical variation. Rather it is due to the combination of the lexical variation with the requirement to make the distinction between the theme and optional cause argument, which cannot be handled appropriately by the small set of GR paths employed.

The contribution of anaphora resolution to our system is limited as it relies on the argument extraction stage which, apart from introducing noise, is geared towards maintaining high precision. Overall, it contributes 22 additional events on the development set, of which 14 out of 16 are correct non-regulation events. Of the remaining 6 regulation events only 2 were correct. Similar trends were observed on the test data.

7 Conclusions - Future work

We described an almost unsupervised approach for the BioNLP09 shared task on biomedical event extraction which requires only a dictionary of verbs and a set of argument extraction rules. Ignoring trigger spans, the performance of the approach is parser-dependent and while we used a domain-independent parser in our experiments we also want to explore the benefits of using an adapted one.

The main weakness of our approach is the handling of events with multiple arguments and the distinctions between them, which are difficult to deal with using simple unlexicalized rules. In our future work we intend to explore semi-supervised approaches that allow us to acquire more complex rules efficiently.

References

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.

Kevin B. Cohen, Martha Palmer, and Lawrence Hunter. 2008. Nominalization and alternations in biomedical language. *PLoS ONE*, 3(9).

Aria Haghighi, Kristina Toutanova, and Chris Manning. 2005. A Joint Model for Semantic Role Labeling. In *Proceedings of CoNLL-2005: Shared Task.*

Anna Korhonen, Yuval Krymolowski, and Nigel Collier. 2008. The choice of features for classification of verbs in biomedical texts. In *Proceedings of Coling*.

Ian Lewin. 2007. BaseNPs that contain gene names: domain specificity and genericity. In *Proceedings of the ACL workshop BioNLP: Biological, translational, and clinical language processing*, pages 163–170.

A Markov Logic Approach to Bio-Molecular Event Extraction

Sebastian Riedel[*†] **Hong-Woo Chun**[*†] **Toshihisa Takagi**[*¶] **Jun'ichi Tsujii**[†‡§]
[*]Database Center for Life Science, Research Organization of Information and System, Japan
[†]Department of Computer Science, University of Tokyo, Japan
[¶]Department of Computational Biology, University of Tokyo, Japan
[‡]School of Informatics, University of Manchester, UK
[§]National Centre for Text Mining, UK
{sebastian,chun,takagi}@dbcls.rois.ac.jp
tsujii@is.s.u-tokyo.ac.jp

Abstract

In this paper we describe our entry to the BioNLP 2009 Shared Task regarding biomolecular event extraction. Our work can be described by three design decisions: (1) instead of building a pipeline using local classifier technology, we design and learn a joint probabilistic model over events in a sentence; (2) instead of developing specific inference and learning algorithms for our joint model, we apply Markov Logic, a general purpose Statistical Relation Learning language, for this task; (3) we represent events as relational structures over the tokens of a sentence, as opposed to structures that explicitly mention abstract event entities. Our results are competitive: we achieve the 4th best scores for task 1 (in close range to the 3rd place) and the best results for task 2 with a 13 percent point margin.

1 Introduction

The continuing rapid development of the Internet makes it very easy to quickly access large amounts of data online. However, it is impossible for a single human to read and comprehend a significant fraction of the available information. Genomics is not an exception, with databases such as MEDLINE storing a vast amount of biomedical knowledge.

A possible way to overcome this is information extraction (IE) based on natural language processing (NLP) techniques. One specific IE sub-task concerns the extraction of molecular events that are mentioned in biomedical literature. In order to drive forward research in this domain, the BioNLP Shared task 2009 (Kim et al., 2009) concerned the extraction of such events from text. In the course of the shared task the organizers provided a training/development set of abstracts for biomedical papers, annotated with the mentioned events. Participants were required to use this data in order to engineer a event predictor which was then evaluated on unseen test data.

The shared task covered three sub-tasks. The first task concerned the extraction of events along with their clue words and their main arguments. Figure 1 shows a typical example. The second task was an extension of the first one, requiring participants to not only predict the core arguments of each event, but also the cellular locations the event is associated with in the text. The events in this task were similar in nature to those in figure 1, but would also contain arguments that are neither events nor proteins but cellular location terms. In contrast to the protein terms, cellular location terms were not given as input and had to be predicted, too. Finally, for task 3 participants were asked to extract negations and speculations regarding events. However, in our work we only tackled Task 1 and Task 2, and hence we omit further details on Task 3 for brevity.

Our approach to biomedical event extraction is inspired by recent work on Semantic Role Labelling (Meza-Ruiz and Riedel, 2009; Riedel and Meza-Ruiz, 2008) and can be characterized by three decisions that we will illustrate in the following. First, we do not build a pipelined system that first predicts event clues and cellular locations, and then relations between these; instead, we design and learn a **joint** discriminative model of the complete event structure for a given sentence. This allows us to incorporate global correlations between decisions in a principled fashion. For example, we know that any event that has arguments which itself are events (such as the positive regulation event in figure 1) has to be a regulation event. This means that when we make the decision about the type of an event (e.g., in the first step of a classification pipeline) *independently* from the decisions about its arguments and their type, we run the risk of violating this constraint. However, in a joint model this can be easily avoided.

Our second design choice is the following: instead of designing and implementing specific inference and training methods for our structured model, we use **Markov Logic**, a Statistical Relational Learning language, and define our global model declaratively. This simplified the implementation of our system significantly, and allowed us to construct a very competitive event extractor in three person-months. For example, the above observation is captured by the simple formula:

$$eventType\,(e,t) \wedge role\,(e,a,r) \wedge event\,(a) \Rightarrow regType\,(t) \quad (1)$$

Finally, we represent event structures as **relational structures over tokens** of a sentence, as opposed to structures that explicitly mention abstract event entities (compare figure 1 and 2). The reason is as follows. Markov Logic, for now, is tailored to *link prediction* problems where we may make inferences about the existence of relations between given entities. However, when the identity and number of objects of our domain is unknown, things become more complicated. By mapping to relational structure over grounded text, we also show a direct connection to recent formulations of Semantic Role Labelling which may be helpful in the future.

The remainder of this paper is organized as follows: we will first present the preprocessing steps we perform (section 2), then the conversion to a link prediction problem (section 3). Subsequently, we will describe Markov Logic (section 4) and our Markov Logic Network for event extraction (section 5). Finally, we present our results (in section 6) and conclude (section 7).

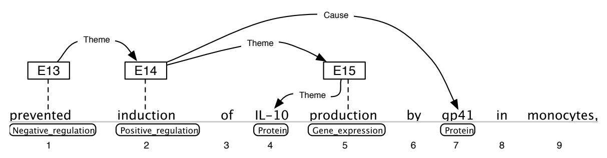


Figure 1: Example gold annotation for task 1 of the shared task.

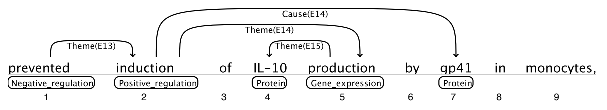


Figure 2: Link Prediction version of the events in figure 1.

2 Preprocessing

The original data format provided by the shared task organizers consists of (a) a collection biomedical abstracts, and (b) standoff annotation that describes the proteins, events and sites mentioned in these abstracts. The organizers also provided a set of dependency and constituent parses for the abstracts. Note that these parses are based on a different tokenisation of the text in the abstracts.

In our first preprocessing step we convert the standoff annotation in the original data to standoff annotation for the tokenisation used in the parses. This allows us to formulate our probabilistic model in terms of one consistent tokenisation (and be able to speak of token instead of character offsets). Then we we retokenise the input text (for the parses) according the protein boundaries that were given in the shared task data (in order to split strings such as "p50/p55"). Finally, we use this tokenisation to once again adapt the stand-off annotation (using the previously adapted version as input).

3 Link Prediction Representation

As we have mentioned earlier, before we learn and apply our Statistical Relational Model, we convert the task to link prediction over a sequence of tokens. In the following we will present this transformation in detail.

To simplify our later presentation we will first introduce a formal representation of the events, proteins and locations mentioned in a sentence. Let us simply identify both proteins and cellular location entities with their token position in the sentence. Furthermore, let us describe an event e as a tuple (i, t, A) where i is the token position of the clue word of e and t is the event type of e; A is a set of labelled arguments (a, r) where each a is either a protein, location or event, and r is the role a plays with respect to e. We will identify the set of all proteins, locations and events for a sentence with P, L and E, respectively.

For example, in figure 1 we have $P = \{4, 7\}$, $L = \emptyset$ and $E = \{e_{13}, e_{14}, e_{15}\}$ with

$$\begin{aligned} e_{15} &= (5, \text{gene\_expr}, \{(4, \text{Theme})\}) \\ e_{14} &= (2, \text{pos\_reg}, \{(e_{15}, \text{Theme}), (7, \text{Cause})\}) \\ e_{13} &= (1, \text{neg\_reg}, \{(e_{14}, \text{Theme})\}) \end{aligned}$$

3.1 Events to Links

As we mentioned in section 1, Markov Logic (or its interpreters) are not yet able to deal with cases where the number and identity of entities is unknown, while relations/links between known objects can be readily modelled. In the following we will therefore present a mapping of an event structure E to a labelled relation over tokens. Essentially, we project E to a pair (L, C) where L is a set of labelled token-to-token links (i, j, r), and C is a set of labelled event clues (i, t). Note that this mapping has another benefit: it creates a "predicate-argument" structure very similar to most recent formulations of Semantic Role Labelling (Surdeanu et al., 2008). Hence it may be possible to re-use or adapt the successful approaches in SRL in order to improve bio-molecular event extraction. Since our approach is inspired by the Markov Logic role labeller in (Riedel and Meza-Ruiz, 2008), this work can be seen as an attempt in this direction.

For a sentence with given P, L and E, algorithm 1 presents our mapping from E to (L, C). For brevity we omit a more detailed description of the algorithm. Note that for our running example *eventsToLinks* would return

$$C = \{(1, \text{neg\_reg}), (2, \text{pos\_reg}), (5, \text{gene\_expr})\} \quad (2)$$

Algorithm 1 Event to link conversion

```
/* returns all clues $C$ and links $L$ given
   by the events in $E$ */
function $eventsToLinks(E)$:
$C \leftarrow \emptyset$, $L \leftarrow \emptyset$
for each event $(i, t, A) \in E$ do
  $C \leftarrow C \cup \{(i, t)\}$
  for each argument $(a, r) \in A$ do
    if $a$ is an event $(i', t', A')$ do
      $L \leftarrow L \cup \{(i, i', r)\}$ with $a = (i', t', A')$
    else
      $L \leftarrow L \cup \{(i, a, r)\}$
return $(C, L)$
```

and

$$\begin{aligned} L = \; & \{(1, 2, \text{Theme}), (2, 5, \text{Theme}), \\ & (2, 7, \text{Cause}), (5, 4, \text{Theme})\}. \end{aligned} \quad (3)$$

3.2 Links to Events

The link-based representation allows us to simplify the design of our Markov Logic Network. However, after we applied the MLN to our data, we still need to transform this representation back to an event structure (in order to use or evaluate it). This mapping is presented in algorithm 2 and discussed in the following. Note that we expect the relational structure L to be cycle free. We again omit a detailed discussion of this algorithm. However, one thing to notice is the special treatment we give to binding events. Roughly speaking, for the binding event clue c we create an event with all arguments of c in L. For a non-binding event clue c we first collect all roles for c, and then create one event per assignment of argument tokens to these roles.

If we would re-convert C and L from equation 2 and 3, respectively, we could return to our original event structure in figure 1. However, converting back and forth is not loss-free in general. For example, if we have a non-binding event in the original E set with two arguments A and B with the same role Theme, the round-trip conversion would generate two events: one with A as Theme and one with B as Theme.

4 Markov Logic

Markov Logic (Richardson and Domingos, 2006) is a Statistical Relational Learning language

Algorithm 2 link to event conversion. Assume: no cycles; tokens can only be one of protein, site or event; binding events have only protein arguments.

```
/* returns all events $E$ specified
   by clues $C$ and links $L$ */
function $linksToEvents(C, L)$
  return $\bigcup_{(i,t)\in C} resolve(i, C, L)$

/* returns all events for
   the given token $i$ */
function $resolve(i, C, L)$
  if no $t$ with $(i,t) \in C$ return $\{i\}$
  $t \leftarrow type(i, C)$
  if $t =$ binding return $\{(i,t,A)\}$ with
    $A = \{(a,r) \mid (i,a,r) \in L\}$
  $R_i \leftarrow \{r' \mid \exists a : (i,a,r) \in L\}$
  for each role $r \in R_i$ do
    $A_r \leftarrow \{a \mid (i,a,r) \in L\}$
    $B_r \leftarrow \bigcup_{a\in A_r} \{(resolve(a), r)\}$
  return $\bigcup_{A\in expand(B_{r_1},\ldots,B_{r_n})} \{(i,t,A)\}$

/* returns all possible argument
   sets for $B_{r_1},\ldots,B_{r_n}$ */
function $expand(B_{r_1},\ldots,B_{r_n})$
  if $n = 1$ return $B_{r_n}$
  return
  $\bigcup_{a\in B_{r_1}} \bigcup_{A\in expand(B_{r_2},\ldots,B_{r_n})} \{(a,r_1)\} \cup A$
```

based on First Order Logic and Markov Networks. It can be seen as a formalism that extends First Order Logic to allow formulae that can be violated with some penalty. From an alternative point of view, it is an expressive template language that uses First Order Logic formulae to instantiate Markov Networks of repetitive structure.

Let us introduce Markov Logic by considering the event extraction task (as relational structure over tokens as generated by algorithm 1). In Markov Logic we can model this task by first introducing a set of logical predicates such as *eventType(Token,Type)*, *role(Token,Token,Role)* and *word(Token,Word)*. Then we specify a set of weighted first order formulae that define a distribution over sets of ground atoms of these predicates (or so-called *possible worlds*). Note that we will refer predicates such as *word* as *observed* because they are known in advance. In contrast, *role* is *hidden* because we need to infer its ground atoms at test time.

Ideally, the distribution we define with these weighted formulae assigns high probability to possible worlds where events are correctly identified and a low probability to worlds where this is not the case. For example, in our running example a suitable set of weighted formulae would assign a higher probability to the world

$$\{word(1, \text{prevented}), eventType(1, \text{neg\_reg}), role(1,2,\text{Theme}), event(2), \ldots\}$$

than to the world

$$\{word(1, \text{prevented}), eventType(1, \text{binding}), role(1,2,\text{Theme}), event(2), \ldots\}$$

In Markov Logic a set of weighted first order formulae is called a *Markov Logic Network* (MLN). Formally speaking, an MLN M is a set of pairs (ϕ, w) where ϕ is a first order formula and w a real weigh t. M assigns the probability

$$p(\mathbf{y}) = \frac{1}{Z} \exp\left(\sum_{(\phi,w)\in M} w \sum_{\mathbf{c}\in C^{\phi}} f_{\mathbf{c}}^{\phi}(\mathbf{y})\right) \quad (4)$$

to the possible world $\mathbf{y}$. Here C^{ϕ} is the set of all possible bindings of the free variables in ϕ with the constants of our domain. $f_{\mathbf{c}}^{\phi}$ is a feature function that returns 1 if in the possible world $\mathbf{y}$ the *ground formula* we get by replacing the free variables in ϕ by the constants in the binding $\mathbf{c}$ is true and 0 otherwise. Z is a normalisation constant.

4.1 Inference and Learning

Assuming that we have an MLN, a set of weights and a given sentence, we need to predict the choice of event clues and roles with maximal *a posteriori* probability (MAP). To this end we apply a method that is both exact and efficient: Cutting Plane Inference Riedel (2008, CPI) with Integer Linear Programming (ILP) as *base solver*.

In order to learn the weights of the MLN we use the 1-best MIRA Crammer and Singer (2003) Online Learning method. As MAP inference method that is applied in the inner loop of the online learner we apply CPI, again with ILP as base solver. The loss function for MIRA is a weighted sum $FP + \alpha FN$ where FP is the number of false positives, FN the number of false negatives and $\alpha = 0.01$.

5 Markov Logic Network for Event Extraction

We define four hidden predicates our task: *event(i)* indicates that there is an event with clue word i; *eventType(i,t)* denotes that at token i there is an event with type t; *site(i)* denotes a cellular location mentioned at token i; *role(i,j,r)* indicates that token i has the argument j with role r. In other words, the four hidden predicates represent the set of sites L (via *site*), the set of event clues C (via *event* and *eventType*) and the set of links L (via *role*) presented in section 3.

There are numerous observed predicates we use. Firstly, the provided information about protein mentions is captured by the predicate *protein(i)*, indicating there is a protein mention ending at token i. We also describe event types and roles in more detail: *regType(*t) holds for an event type t iff it is a regulation event type; *task1Role*(r) and *task2Role*(r) hold for a role r if is a role of task 1 (Theme, Cause) or task 2 (Site, CSite, etc.).

Furthermore, we use predicates that describe properties of tokens (such as the word or stem of a token) and token pairs (such as the dependency between two tokens); this set is presented in table 1. Here the *path* and *pathNL* predicates may need some further explanation. When *path(i,j,p,parser)* is true, there must be a labelled dependency path p between i and j according to the parser *parser*. For example, in figure 1 we will observe *path*(1,5,dobj↓prep_of↓,mcclosky-charniak). pathNL just omits the dependency labels, leading to *path*(1,5,↓↓,mcclosky-charniak) for the same example.

We use two parses per sentence: the outputs of a self-trained reranking parser Charniak and Johnson (2005); McClosky and Charniak (2008) and a CCG parser (Clark and Curran, 2007), provided as part of the shared task dataset. As dictionaries we use a collection of cellular location terms taken from the Genia event corpus (Kim et al., 2008), a small handpicked set of event triggers and a list of English stop words.

Predicate	Description
word(i,w)	Token i has word w.
stem(i,s)	i has (Porter) stem s.
pos(i,p)	i has POS tag p.
hyphen(i,w)	i has word w after last hyphen.
hyphenStem(i,s)	i has stem s after last hyphen.
dict(i,d)	i appears in dictionary d.
genia(i,p)	i is event clue in the Genia corpus with precision p.
dep(i,j,d,parser)	i is head of token j with dependency d according to parser *parser*.
path(i,j,p,parser)	Labelled Dependency path according to parser *parser* between tokens i and j is p.
pathNL(i,j,p,parser)	Unlabelled dependency path according to parser p between tokens i and j is *path*.

Table 1: Observable predicates for token and token pair properties.

5.1 Local Formulae

A formula is *local* if its groundings relate any number of observed ground atoms to exactly one hidden ground atom. For example, the grounding

$$dep\,(1, 2, \mathrm{dobj}, \mathrm{ccg}) \wedge word\,(1, \mathrm{prevented}) \Rightarrow eventType\,(2, \mathrm{pos\_reg}) \quad (5)$$

of the local formula

$$dep(h, i, d, parser) \wedge word\,(h, +w) \Rightarrow eventType(i, +t) \quad (6)$$

connects a single hidden *eventType* ground atom with an observed *word* and *dep* atom. Note that the "+" prefix for variables indicates that there is a different weight for each possible pair of word and event type (w, t).

5.1.1 Local Entity Formulae

The local formulae for the hidden *event/1* predicate can be summarized as follows. First, we add a *event* (i) formula that postulates the existence of an event for each token. The weight of this formulae serves as a general bias for or against the existence of events.

Next, we add one formula

$$T\,(i, +t) \Rightarrow \mathit{event}\,(i) \tag{7}$$

for each "simple token property" predicate T in table 1 (those in the first section of the table). For example, when we plug in *word* for T we get a formula that encourages or discourages the existence of an event token based on the word form of the current token: $\mathit{word}\,(i, +t) \Rightarrow \mathit{event}\,(i)$.

We also add the formula

$$\mathit{genia}\,(i, p) \Rightarrow \mathit{event}\,(i) \tag{8}$$

and multiply the feature-weight product for each of its groundings with the precision p. This is corresponds to so-called real-valued feature functions, and allows us to incorporate probabilities and other numeric quantities in a principled fashion.

Finally, we add a version of formula 6 where we replace *eventType(i,t)* with *event(i)*.

For the cellular location *site* predicate we use exactly the same set of formulae but replace every occurrence of *event(i)* with *site(i)*. This demonstrates the ease with which we could tackle task 2: apart from a small set of global formulae we introduce later, we did not have to do more than copy one file (the event model file) and perform a search-and-replace. Likewise, in the case of the *eventType* predicate we simply replace *event(i)* with *eventType(i,+t)*.

5.1.2 Local Link Formulae

The local formulae for the *role/3* predicate are different in nature because they assess two tokens and their relation. However, the first formula does look familiar: $\mathit{role}\,(i, j, +r)$. This formula captures a (role-dependent) bias for the existence of a role between any two tokens.

The next formula we add is

$$\mathit{dict}\,(i, +d_i) \wedge \mathit{dict}\,(j, +d_j) \Rightarrow \mathit{role}\,(i, j, +r) \tag{9}$$

and assesses each combination of dictionaries that the event and argument token are part of. Furthermore, we add the formula

$$\mathit{path}\,(i, j, +p, +\mathit{parser}) \Rightarrow \mathit{role}\,(i, j, +r) \tag{10}$$

that relates the dependency path between two tokens i and j with the role that j plays with respect to i. We also add an unlabelled version of this formula (using *pathNL* instead of *path*). Finally, we add a formula

$$P\,(i, j, +p, +\mathit{parser}) \wedge T\,(i, +t) \Rightarrow \\ \mathit{role}\,(i, j, +r) \tag{11}$$

for each P in {*path,pathNL*} and T in {*word,stem,pos,dict,protein*}. Note that for T=*protein* we replace $T\,(i, +t)$ with $T\,(i)$.

5.2 Global Formulae

Global formulae relate two or more hidden ground atoms. For example, the formula in equation 1 is global. While local formulae can be used in any conventional classifier (in the form of feature functions conditioned only on the input data) this does not hold for global ones. We could enforce global constraints such as the formula in equation 1 by building up structure incrementally (e.g. start with one classifier for events and sites, and then predict roles between events and arguments with another). However, this does not solve the typical chicken-and-egg problem: evidence for possible arguments could help us to predict the existence of event clues, and evidence for events help us to predict arguments. By contrast, global formulae can capture this type of correlation very naturally.

Table 2 shows the global formulae we use. We divide them into three parts. The first set of formulae (CORE) ensures that *event* and *eventType* atoms are consistent. In all our experiments we will always include all CORE formulae; without them we might return meaningless solutions that have events with no event types, or types without events.

The second set of formulae (VALID) consist of CORE and formulae that ensure that the link structure represents a valid set of events. For example, this includes formula 12 that enforces each event to have at least one theme.

Finally, FULL includes VALID and two constraints that are not strictly necessary to enforce valid event structures. However, they do help us to improve performance. Formula 14 forbids a token to be argument of more than one event. In fact, this formula does not hold all the time, but

#	Formula	Description
1	$event(i) \Rightarrow \exists t.eventType(i,t)$	If there is an event there should be an event type.
2	$eventType(i,t) \Rightarrow event(i)$	If there is an event type there should be an event.
3	$eventType(i,t) \wedge t \neq o \Rightarrow \neg eventType(i,o)$	There cannot be more than one event type per token.
4	$\neg site(i) \vee \neg event(i)$	A token cannot be both be event and site.
5	$role(i,j,r) \Rightarrow event(i)$	If j plays the role r for i then i has to be an event.
6	$role(i,j,r_1) \wedge r_1 \neq r_2 \Rightarrow \neg role(i,j,r_2)$	There cannot be more than one role per argument.
7	$eventType(e,t) \wedge role(e,a,r) \wedge event(a) \Rightarrow regType(t)$	Only reg. type events can have event arguments.
9	$role(i,j,r) \wedge taskOne(r) \Rightarrow event(j) \vee protein(j)$	For task 1 roles arguments must be proteins or events
10	$role(i,j,r) \wedge taskTwo(r) \Rightarrow site(j)$	Task 2 arguments must be cellular locations ($site$).
11	$site(j) \Rightarrow \exists i,r.role(i,j,r) \wedge taskTwo(r)$	Sites are always associated with an event.
12	$event(i) \Rightarrow \exists j.role(i,j,\mathrm{Theme})$	Every events need a theme.
13	$eventType(i,t) \wedge \neg allowed(t,r) \Rightarrow \neg role(i,j,r)$	Certain events may not have certain roles.
14	$role(i,j,r_1) \wedge k \neq i \Rightarrow \neg role(k,j,r_2)$	A token cannot be argument of more than one event.
15	$j < k \wedge i < j \wedge role(i,j,r_1) \Rightarrow \neg role(i,k,r_2)$	No inside outside chains.

Table 2: All three sets of global formulae used: CORE (1-3), VALID (1-13), FULL (1-15).

by adding it we could improve performance. Formula 15 is our answer to a type of event "chain" that earlier models would tend to produce.

Note that all formulae but formula 15 are deterministic. This amounts to giving them a very high/infinite weight in advance (and not learning it during training).

6 Results

In table 3 we can see our results for task 1 and 2 of the shared task. The measures we present here correspond to the "approximate span, approximate recursive match" criterion that counts an event as correctly predicted if all arguments are extracted and the event clue tokens approximately match the gold clue tokens. For more details on this metric we refer the reader to the shared task overview paper.

To put our results into context: for task 1 we reached the 4th place among 20 participants, are in close range to place 2 and 3, and significantly outperform the 5th best entry. Moreover, we had highest scoring scores for task 2 with a 13% margin to the runner-up. Using both training and development set for training (as allowed by the task organisers), our task 1 score rises to 45.1, slightly higher than the score of the current third.

In terms of accuracy across different event types our model performs worse for binding, regulation type and transcription events. Binding events are inherently harder to correctly extract because they often have multiple core arguments while other non-regulation events have only one; just missing one of the binding arguments will lead to an event that is considered as error with no partial credit given. If we would give credit for binding with partially correct arguments our F-score for binding events would rise to 49.8.

One reason why regulation events are difficult to extract is the fact that they often have arguments which themselves are events, too. In this case our recall is bound by the recall for argument events because we can never find a regulation event if we cannot predict the argument event. Note that we are still unsure about transcription events, in particular because we observe 49% F-score for such events in the development set.

How does our model benefit from the global formulae we describe in section 5 (and which represent one of the core benefits of a Markov Logic approach)? To evaluate this we compare our FULL model with CORE and VALID from table 2. Note that because the evaluation interface rejects invalid event structures, we cannot use the evaluation metrics of the shared task. Instead we use table 4 to present an evaluation in terms of ground atom F1-score for the hidden predicates of our model. This amounts to a per-

	Task 1			Task 2		
	R	P	F	R	P	F
Loc	37.9	88.0	53.0	32.8	76.0	45.8
Bind	23.1	48.2	31.2	22.4	47.0	30.3
Expr	63.0	75.1	68.5	63.0	75.1	68.5
Trans	16.8	29.9	21.5	16.8	29.9	21.5
Cata	64.3	81.8	72.0	64.3	81.8	72.0
Phos	78.5	77.4	77.9	69.1	70.1	69.6
Total	48.3	68.9	56.8	46.8	67.0	55.1
Reg	23.7	40.8	30.0	22.3	38.5	28.2
Pos	26.8	42.8	32.9	26.7	42.3	32.7
Neg	27.2	40.2	32.4	26.1	38.6	31.2
Total	26.3	41.8	32.3	25.8	40.8	31.6
Total	36.9	55.6	44.4	35.9	54.1	43.1

Table 3: (R)ecall, (P)recision, and (F)-Score for task 1 and 2 in terms of event types.

role, per-site and per-event-clue evaluation. The numbers here will not directly correspond to actual scores, but generally we can assume that if we do better in our metrics, we will likely have better scores.

In table 4 we notice that ensuring consistency between all predicates has a significant impact on the performance across the board (see the VALID results). Furthermore, when adding extra formulae that are not strictly necessary for consistency, but which encourage more likely event structure, we again see significant improvements (see FULL results). Interestingly, although the extra formulae only directly consider *role* atoms, they also have a significant impact on event and particularly site extraction performance. This reflects how in a joint model decisions which would appear in the end of a traditional pipeline (e.g., extracting roles for events) can help steps that would appear in the beginning (extracting events and sites).

For the about 7500 sentences in the training set we need about 3 hours on a MacBook Pro with 2.8Ghz and 4Gb RAM to learn the weights of our MLN. This allowed us to try different sets of formulae in relatively short time.

7 Conclusion

Our approach the BioNLP Shared Task 2009 can be characterized by three decisions: (a) jointly modelling the complete event structure for a given sentence; (b) using Markov Logic as general purpose-framework in order to implement our joint model; (c) framing the problem as a link prediction problem between tokens of a sentence.

	CORE	VALID	FULL
eventType	52.8	63.2	64.3
role	44.0	53.5	55.7
site	42.0	46.0	51.5
Total	50.7	60.1	61.9

Table 4: Ground atom F-scores for global formulae.

Our results are competitive: we reach the 4th place in task 1 and the 1st place for task 2 (with a 13% margin). Furthermore, the declarative nature of Markov Logic helped us to achieve these results with a moderate amount of engineering. In particular, we were able to tackle task 2 by copying the local formulae for event prediction, and adding three global formulae (4, 10 and 11 in table 2). Finally, our system was fast to train (3 hours) . This greatly simplified the search for good sets of formulae.

We have also shown that global formulae significantly improve performance in terms of event clue, site and argument prediction. While a similar effect may be possible with reranking architectures, we believe that in terms of implementation efforts our approach is at least as simple. In fact, our main effort lied in the conversion to link prediction, not in learning or inference. In future work we will therefore investigate means to extend Markov Logic (interpreter) in order to directly model event structure.

Acknowledgements

We thank Dr. Chisato Yamasaki and Dr. Tadashi Imanishi, BIRC, AIST, for their help. This work is supported by the Integrated Database Project (MEXT, Japan), the Grant-in-Aid for Specially Promoted Research (MEXT, Japan) and the Genome Network Project (MEXT, Japan).

References

Charniak, Eugene and Mark Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL' 05)*. pages 173–180.

Clark, Stephen and James R. Curran. 2007. Wide-coverage efficient statistical parsing with ccg and log-linear models. *Comput. Linguist.* 33(4):493–552.

Crammer, Koby and Yoram Singer. 2003. Ultraconservative online algorithms for multiclass problems. *Journal of Machine Learning Research* 3:951–991.

Kim, Jin D., Tomoko Ohta, and Jun'ichi Tsujii. 2008. Corpus annotation for mining biomedical events from literature. *BMC Bioinformatics* 9(1).

Kim, Jin-Dong, Tomoko Ohta, Sampo Pyysalo, Yoshinobu Kano, and Jun'ichi Tsujii. 2009. Overview of bionlp'09 shared task on event extraction. In *Proceedings of Natural Language Processing in Biomedicine (BioNLP) NAACL 2009 Workshop*. To appear.

McClosky, David and Eugene Charniak. 2008. Self-training for biomedical parsing. In *Proceedings of the 46rd Annual Meeting of the Association for Computational Linguistics (ACL' 08)*.

Meza-Ruiz, Ivan and Sebastian Riedel. 2009. Jointly identifying predicates, arguments and senses using markov logic. In *Joint Human Language Technology Conference/Annual Meeting of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL '09)*.

Richardson, Matt and Pedro Domingos. 2006. Markov logic networks. *Machine Learning* 62:107–136.

Riedel, Sebastian. 2008. Improving the accuracy and efficiency of map inference for markov logic. In *Proceedings of the 24th Annual Conference on Uncertainty in AI (UAI '08)*.

Riedel, Sebastian and Ivan Meza-Ruiz. 2008. Collective semantic role labelling with markov logic. In *Proceedings of the 12th Conference on Computational Natural Language Learning (CoNLL' 08)*. pages 193–197.

Surdeanu, Mihai, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. 2008. The CoNLL-2008 shared task on joint parsing of syntactic and semantic dependencies. In *Proceedings of the 12th Conference on Computational Natural Language Learning (CoNLL-2008)*.

High-precision biological event extraction with a concept recognizer

K. Bretonnel Cohen*, Karin Verspoor*, Helen L. Johnson, Chris Roeder,
Philip V. Ogren, William A. Baumgartner Jr., Elizabeth White, Hannah Tipney, and Lawrence Hunter
Center for Computational Pharmacology
University of Colorado Denver School of Medicine
PO Box 6511, MS 8303, Aurora, CO 80045 USA
kevin.cohen@gmail.com, karin.verspoor@ucdenver.edu, helen.linguist@gmail.com,
chris.roeder@ucdenver.edu, philip@ogren.info, william.baumgartner@ucdenver.edu,
elizabeth.white@colorado.edu, hannah.tipney@ucdenver.edu, larry.hunter@ucdenver.edu

Abstract

We approached the problems of event detection, argument identification, and negation and speculation detection as one of concept recognition and analysis. Our methodology involved using the OpenDMAP semantic parser with manually-written rules. We achieved state-of-the-art precision for two of the three tasks, scoring the highest of 24 teams at precision of 71.81 on Task 1 and the highest of 6 teams at precision of 70.97 on Task 2.

The OpenDMAP system and the rule set are available at bionlp.sourceforge.net.

*These two authors contributed equally to the paper.

1 Introduction

We approached the problem of biomedical event recognition as one of concept recognition and analysis. Concept analysis is the process of taking a textual input and building from it an abstract representation of the concepts that are reflected in it. Concept recognition can be equivalent to the named entity recognition task when it is limited to locating mentions of particular semantic types in text, or it can be more abstract when it is focused on recognizing predicative relationships, e.g. events and their participants.

2 BioNLP'09 Shared Task

Our system was entered into all three of the BioNLP'09 (Kim et al., 2009) shared tasks:

- **Event detection and characterization** This task requires recognition of 9 basic biological events: gene expression, transcription, protein catabolism, protein localization, binding, phosphorylation, regulation, positive regulation and negative regulation. It requires identification of the core THEME and/or CAUSE participants in the event, i.e. the protein(s) being produced, broken down, bound, regulated, etc.

- **Event argument recognition** This task builds on the previous task, adding in additional arguments of the events, such as the site (protein or DNA region) of a binding event, or the location of a protein in a localization event.

- **Recognition of negations and speculations** This task requires identification of negations of events (e.g. event X did *not* occur), and speculation about events (e.g. We *claim* that event X *should* occur).

3 Our approach

We used the OpenDMAP system developed at the University of Colorado School of Medicine (Hunter et al., 2008) for our submission to the BioNLP '09 Shared Task on Event Extraction. OpenDMAP is an ontology-driven, integrated concept analysis system that supports information extraction from text through the use of patterns represented in a classic form of "semantic grammar," freely mixing text literals, semantically typed basal syntactic constituents, and semantically defined classes of entities. Our approach is to take advantage of the high

Proceedings of the Workshop on BioNLP: Shared Task, pages 50–58,
Boulder, Colorado, June 2009. ©2009 Association for Computational Linguistics

quality ontologies available in the biomedical domain to formally define entities, events, and constraints on slots within events and to develop patterns for how concepts can be expressed in text that take advantage of both semantic and linguistic characteristics of the text. We manually built patterns for each event type by examining the training data and by using native speaker intuitions about likely ways of expressing relationships, similar to the technique described in (Cohen et al., 2004). The patterns characterize the linguistic expression of that event and identify the arguments (participants) of the events according to (a) occurrence in a relevant linguistic context and (b) satisfaction of appropriate semantic constraints, as defined by our ontology. Our solution results in very high precision information extraction, although the current rule set has limited recall.

3.1 The reference ontology

The central organizing structure of an OpenDMAP project is an ontology. We built the ontology for this project by combining elements of several community-consensus ontologies—the Gene Ontology (GO), Cell Type Ontology (CTO), BRENDA Tissue Ontology (BTO), Foundational Model of Anatomy (FMA), Cell Cycle Ontology (CCO), and Sequence Ontology (SO)—and a small number of additional concepts to represent task-specific aspects of the system, such as event trigger words. Combining the ontologies was done with the Prompt plug-in for Protégé.

The ontology included concepts representing each event type. These were represented as frames, with slots for the various things that needed to be returned by the system—the trigger word and the various slot fillers. All slot fillers were constrained to be concepts in some community-consensus ontology. The core event arguments were constrained in the ontology to be of type *protein* from the Sequence Ontology (except in the case of regulation events, where biological events themselves could satisfy the THEME role), while the type of the other event arguments varied. For instance, the ATLOC argument of a gene expression event was constrained to be one of tissue (from BTO), cell type (from CTO), or cellular component (from GO-Cellular Component), while the BINDING argument of a binding event was constrained to be one of binding_site, DNA, domain, or chromosome (all from the SO and all tagged by LingPipe). Table 1 lists the various types.

3.2 Named entity recognition

For proteins, we used the gold standard annotations provided by the organizers. For other semantic classes, we constructed a compound named entity recognition system which consists of a LingPipe GENIA tagging module (LingPipe, (Alias-i, 2008)), and several dictionary look-up modules. The dictionary lookup was done using a component from the UIMA (IBM, 2009; Ferrucci and Lally, 2004) sandbox called the ConceptMapper.

We loaded the ConceptMapper with dictionaries derived from several ontologies, including the Gene Ontology Cellular Component branch, Cell Type Ontology, BRENDA Tissue Ontology, and the Sequence Ontology. The dictionaries contained the names and name variants for each concept in each ontology, and matches in the input documents were annotated with the relevant concept ID for the match. The only modifications that we made to these community-consensus ontologies were to remove the single concept *cell* from the Cell Type Ontology and to add the synonym *nuclear* to the Gene Ontology Cell Component concept *nucleus*.

The protein annotations were used to constrain the text entities that could satisfy the THEME role in the events of interest. The other named entities were added for the identification of non-core event participants for Task 2.

3.3 Pattern development strategies

3.3.1 Corpus analysis

Using a tool that we developed for visualizing the training data (described below), a subset of the gold-standard annotations were grouped by event type and by trigger word type (nominalization, passive verb, active verb, or multiword phrase). This organization helped to suggest the argument structures of the event predicates and also highlighted the variation within argument structures. It also showed the nature of more extensive intervening text that would need to be handled for the patterns to achieve higher recall.

Based on this corpus analysis, patterns were developed manually using an iterative process in which individual patterns or groups of patterns were tested

Table 1: Semantic restrictions on Task 2 event arguments. CCO = Cell Cycle Ontology, FMA = Foundational Model of Anatomy, other ontologies identified in the text.

Event Type	**Site**	**AtLoc**	**ToLoc**
binding	protein domain (SO), binding site (SO), DNA (SO), chromosome (SO)		
gene expression	gene (SO), biological entity (CCO)	tissue (BTO), cell type (CTO), cellular component (GO)	
localization		cellular component (GO)	cellular component (GO)
phosphorylation	amino acid (FMA), polypeptide region (SO)		
protein catabolism	cellular component (GO)		
transcription	gene (SO), biological entity (CCO)		

on the training data to determine their impact on performance. Pattern writers started with the most frequent trigger words and argument structures.

3.3.2 Trigger words

In the training data, we were provided annotations of all relevant event types occurring in the training documents. These annotations included a *trigger word* specifying the specific word in the input text which indicated the occurrence of each event. We utilized the trigger words in the training set as anchors for our linguistic patterns. We built patterns around the generic concept of, e.g. an *expression trigger word* and then varied the actual strings that were allowed to satisfy that concept. We then ran experiments with our patterns and these varying sets of trigger words for each event type, discarding those that degraded system performance when evaluated with respect to the gold standard annotations.

Most often a trigger word was removed from an event type trigger list because it was also a trigger word for another event type and therefore reduced performance by increasing the false positive rate. For example, the trigger words "level" and "levels" appear in the training data trigger word lists of gene expression, transcription, and all three regulation event types.

The selection of trigger words was guided by a frequency analysis of the trigger words provided in the task training data. In a post-hoc analysis, we find that a different proportion of the set of trigger words was finally chosen for each different event type. Between 10-20% of the top frequency-ranked trigger words were used for simple event types, with the exception that phosphorylation trigger words were chosen from the top 30%. For instance, for gene expression all of the top 15 most frequent trigger words were used (corresponding to the top 16%). For complex event types (the regulations) better performance was achieved by limiting the list to between 5-10% of the most frequent trigger words.

In addition, variants of frequent trigger words were included. For instance, the nominalization "expression" is the most frequent gene expression trigger word and the verbal inflections "expressed" and "express" are also in the top 20%. The verbal inflection "expresses" is ranked lower than the top 30%, but was nonetheless included as a trigger word in the gene expression patterns.

3.3.3 Patterns

As in our previous publications on OpenDMAP, we refer to our semantic rules as *patterns*. For this task, each pattern has at a minimum an event argument THEME and an event-specific trigger word. For example, $\{phosphorylation\}$:=

$[phosphorylation\_nominalization][Theme]$, where $[phosphorylization\_nominalization]$ represents a trigger word. Both elements are defined semantically. Event THEMEs are constrained by restrictions placed on them in the ontology, as described above.

The methodology for creating complex event patterns such as regulation was the same as for simple events, with the exception that the THEMEs were defined in the ontology to also include biological processes. Iterative pattern writing and testing was a little more arduous because these patterns relied on the success of the simple event patterns, and hence more in-depth analysis was required to perform performance-increasing pattern adjustments. For further details on the pattern language, the reader is referred to (Hunter et al., 2008).

3.3.4 Nominalizations

Nominalizations were very frequent in the training data; for seven out of nine event types, the most common trigger word was a nominalization. In writing our grammars, we focused on these nominalizations. To write grammars for nominalizations, we capitalized on some of the insights from (Cohen et al., 2008). Non-ellided (or otherwise absent) arguments of nominalizations can occur in three basic positions:

- Within the noun phrase, after the nominalization, typically in a prepositional phrase
- Within the noun phrase, immediately preceding the nominalization
- External to the noun phrase

The first of these is the most straightforward to handle in a rule-based approach. This is particularly true in the case of a task definition like that of BioNLP '09, which focused on themes, since an examination of the training data showed that when themes were post-nominal in a prepositional phrase, then that phrase was most commonly headed by *of*.

The second of these is somewhat more challenging. This is because both agents and themes can occur immediately before the nominalization, e.g. *phenobarbital induction* (induction *by* phenobarbital) and *trkA expression* (expression *of* trkA). To decide how to handle pre-nominal arguments, we made use of the data on semantic roles and syntactic position found in (Cohen et al., 2008). That study found that themes outnumbered agents in the prenominal position by a ratio of 2.5 to 1. Based on this observation, we assigned pre-nominal arguments to the theme role.

Noun-phrase-external arguments are the most challenging, both for automatic processing and for human interpreters; one of the major problems is to differentiate between situations where they are present but outside of the noun phrase, and situations where they are entirely absent. Since the current implementation of OpenDMAP does not have robust access to syntactic structure, our only recourse for handling these arguments was through wildcards, and since they mostly decreased precision without a corresponding increase in recall, we did not attempt to capture them.

3.3.5 Negation and speculation

Corpus analysis of the training set revealed two broad categories each for negation and speculation modifications, all of which can be described in terms of the scope of modification.

Negation

Broadly speaking, an event itself can be negated or some aspect of an event can be negated. In other words, the scope of a negation modification can be over the existence of an event (first example below), or over an argument of an existing event (second example).

- *This $\underline{\textit{failure to degrade IkappaBalpha}}$ ...* (PMID 10087185)
- *AP-1 $\underline{\textit{but not NF-IL-6}}$ DNA binding activity ...* (PMID 10233875)

Patterns were written to handle both types of negation. The negation phrases "but not" and "but neither" were appended to event patterns to catch those events that were negated as a result of a negated argument. For event negation, a more extensive list of trigger words was used that included verbal phrases such as "failure to" and "absence of."

The search for negated events was conducted in two passes. Events for which negation cues fall outside the span of text that stretches from argument to event trigger word were handled concurrently with the search for events. A second search was conducted on extracted events for negation cues that fell within the argument to event trigger word span, such as

. . . *IL-2 does not induce I kappa B alpha degradation* (PMID 10092783)

This second pass allowed us to capture one additional negation (6 rather than 5) on the test data.

Speculation

The two types of speculation in the training data can be described by the distinction between "de re" and "de dicto" assertions. The "de dicto" assertions of speculation in the training data are modifications that call into question the degree of known truth of an event, as in

. . . *CTLA-4 ligation did not appear to affect the CD28 - mediated stabilization* (PMID 10029815)

The "de re" speculation address the potential existence of an event rather that its degree of truth. In these cases, the event is often being introduced in text by a statement of intention to study the event, as in

. . . *we investigated CTCF expression . . . [10037138]*

To address these distinct types of speculation, two sets of trigger words were developed. One set consisted largely of verbs denoting research activities, e.g. *research, study, examine investigate,* etc. The other set consisted of verbs and adverbs that denote uncertainty, and included trigger words such as *suggests, unknown,* and *seems*.

3.4 Handling of coordination

Coordination was handled using the OpenNLP constituent parser along with the UIMA wrappers that they provide via their code repository. We chose OpenNLP because it is easy to train a model, it integrates easily into a UIMA pipeline, and because of competitive parsing results as reported by Buyko (Buyko et al., 2006). The parser was trained using 500 abstracts from the beta version of the GENIA treebank and 10 full-text articles from the CRAFT corpus (Verspoor et al., In press). From the constituent parse we extracted coordination structures into a simplified data structure that captures each conjunction along with its conjuncts. These were provided to downstream components. The coordination component achieves an F-score of 74.6% at the token level and an F-score of 57.5% at the conjunct level when evaluated against GENIA. For both measures the recall was higher than the precision by 4% and 8%, respectively.

We utilized the coordination analysis to identify events in which the THEME argument was expressed as a conjoined noun phrase. These were assumed to have a distributed reading and were post-processed to create an individual event involving each conjunct, and further filtered to only include given (A1) protein references. So, for instance, analysis of the sentence in the example below should result in the detection of three separate gene expression events, involving the proteins HLA-DR, CD86, and CD40, respectively.

> NAC was shown to down-regulate the production of cytokines by DC as well as **their surface expression of HLA-DR, CD86 (B7-2), and CD40 molecules** . . . (PMID 10072497)

3.5 Software infrastructure

We took advantage of our existing infrastructure based on UIMA (The Unstructured Information Management Architecture) (IBM, 2009; Ferrucci and Lally, 2004) to support text processing and data analysis.

3.5.1 Development tools

We developed a visualization tool to enable the linguistic pattern writers to better analyze the training data. This tool shows the source text one sentence at a time with the annotated words highlighted. A list following each sentence shows details of the annotations.

3.6 Errors in the training data

In some cases, there were discrepancies between the training data and the official problem definitions. This was a source of problems in the pattern development phase. For example, phosphorylation events are defined in the task definition as having only a THEME and a SITE. However, there were instances in the training data that included both a THEME and a CAUSE argument. When those events were identified by our system and the CAUSE was labelled, they were rejected during a syntactic error check by the test server.

4 Results

4.1 Official Results

We are listed as Team 13. Table 2 shows our results on the official metrics. Our precision was the highest achieved by any group for Task 1 and Task 2, at 71.81 for Task 1 and 70.97 for task 2. Our recalls were much lower and adversely impacted our F-measure; ranked by F-measure, we ranked 19th out of 24 groups.

We noted that our results for the exact match metric and for the approximate match metric were very close, suggesting that our techniques for named entity recognition and for recognizing trigger words are doing a good job of capturing the appropriate spans.

4.2 Other analysis: Bug fixes and coordination handling

In addition to our official results, we also report in Table 3 (see last page) the results of a run in which we fixed a number of bugs. This represents our current best estimate of our performance. The precision drops from 71.81 for Task 1 to 67.19, and from 70.97 for Task 2 to 65.74, but these precisions are still well above the second-highest precisions of 62.21 for Task 1 and 56.87 for Task 2. As the table shows, we had corresponding small increases in our recall to 17.38 and in our F-measure to 27.62 for Task 1, and in our recall to 17.07 and F-measure to 27.10 for Task 2.

We evaluated the effects of coordination handling by doing separate runs with and without this element of the processing pipeline. Compared to our unofficial results, which had an overall F-measure for Task 1 of 27.62 and for Task 2 of 27.10, a version of the system without handling of coordination had an overall F-measure for Task 1 of 24.72 and for Task 2 of 24.21.

4.3 Error Analysis

4.3.1 False negatives

To better understand the causes of our low recall, we performed a detailed error analysis of false negatives using the devtest data. (Note that this section includes a very small number of examples from the devtest data.) We found five major causes of false negatives:

- Intervening material between trigger words and arguments
- Coordination that was not handled by our coordination component
- Low coverage of trigger words
- Anaphora and coreference
- Appositive gene names and symbols

Intervening material For reasons that we detail in the *Discussion* section, we avoided the use of wildcards. This, and the lack of syntactic analysis in the version of the system that we used (note that syntactic analyses *can* be incorporated into an OpenDMAP workflow), meant that if there was text intervening between a trigger word and an argument, e.g. in *to efficiently [express] in developing thymocytes a mutant form of the [NF-kappa B inhibitor]* (PMID 10092801), where the bracketed text is the trigger word and the argument, our pattern would not match.

Unhandled coordination Our coordination system only handled coordinated protein names. Thus, in cases where other important elements of the utterance, such as the trigger word *transcription* in *transcription and subsequent synthesis and secretion of galectin-3* (PMID 8623933) were in coordinated structures, we missed the relevant event arguments.

Low coverage of trigger words As we discuss in the *Methods* section, we did not attempt to cover all trigger words, in part because some less-frequent trigger words were involved in multiple event types, in part because some of them were extremely low-frequency and we did not want to overfit to the training data, and in part due to the time constraints of the shared task.

Anaphora and coreference Recognition of some events in the data would require the ability to do anaphora and coreference resolution. For example, in *Although 2 early lytic transcripts, [BZLF1] and [BHRF1], were also detected in 13 and 10 cases, respectively, the lack of ZEBRA staining in any case indicates that these lytic transcripts are most likely*

	Tasks 1 and 3					Task 2		
Event class	GS	answer	R	P	F	R	P	F
Localization	174 (18)	18 (18)	10.34	100.00	18.75	9.77	94.44	17.71
Binding	347 (44)	110 (44)	12.68	40.00	19.26	12.32	39.09	18.74
Gene expression	722 (263)	306 (263)	36.43	85.95	51.17	36.43	85.95	51.17
Transcription	137 (18)	20 (18)	13.14	90.00	22.93	13.14	90.00	22.93
Protein catabolism	14 (4)	6 (4)	28.57	66.67	40.00	28.57	66.67	40.00
Phosphorylation	135 (30)	30 (30)	22.22	100.00	36.36	20.14	93.33	33.14
EVENT TOTAL	1529 (377)	490 (377)	24.66	76.94	37.35	24.30	76.12	36.84
Regulation	291 (9)	19 (9)	3.09	47.37	5.81	3.08	47.37	5.79
Positive regulation	983 (32)	65 (32)	3.26	49.23	6.11	3.24	49.23	6.08
Negative regulation	379 (10)	22 (10)	2.64	45.45	4.99	2.37	40.91	4.49
REGULATION TOTAL	1653 (51)	106 (51)	3.09	48.11	5.80	3.02	47.17	5.67
Negation	227 (4)	76 (4)	1.76	5.26	2.64			
Speculation	208 (14)	105 (14)	6.73	13.33	8.95			
MODIFICATION TOTAL	435 (18)	181 (18)	4.14	9.94	5.84			
ALL TOTAL	3182 (428)	596 (428)	13.45	71.81	22.66	13.25	70.97	22.33

Table 2: Official scores for Tasks 1 and 2, and modification scores only for Task 3, from the approximate span matching/approximate recursive matching table. GS = gold standard (true positives) (given for Tasks 1/3 only), answer = all responses (true positives) (given for tasks 1/3 only), R = recall, P = precision, F = F-measure. All results are as calculated by the official scoring application.

[expressed] by rare cells in the biopsies entering lytic cycle (PMID 8903467), where the bracketed text is the arguments and the trigger word, the syntactic object of the verb is the anaphoric noun phrase *these lytic transcripts*, so even with the addition of a syntactic component to our system, we still would not have recognized the appropriate arguments without the ability to do anaphora resolution.

Appositives The annotation guidelines for proteins apparently specified that when a gene name was present in an appositive with its symbol, the symbol was selected as the gold-standard argument. For this reason, in examples like *[expression] of Fas ligand [FasL]* (PMID 10092076), where the bracketed text is the trigger word and the argument, the gene name constituted intervening material from the perspective of our patterns, which therefore did not match.

We return to a discussion of recall and its implications for systems like ours in the *Discussion* section.

4.3.2 False positives

Although our overall rate of false positives was low, we sampled 45 false positive events distributed across the nine event types and reviewed them with a biologist.

We noted two main causes of error. The most common was that we misidentified a slot filler or were missing a slot filler completely for an actual event. The other main reason for false positives was when we erroneously identified a (non)event. For example, in *coexpression of NF-kappa B/Rel and Sp1 transcription factors* (PMID 7479915), we mistakenly identified *Sp1 transcription* as an event.

5 Discussion

Our results demonstrate that it is possible to achieve state-of-the art precision over a broad range of tasks and event types using our approach of manually constructed, ontologically typed rules—our precision of 71.81 on Task 1 was ten points higher than the second-highest precision (62.21), and our precision of 70.97 on Task 2 was 14 points higher than the second-highest precision (56.87). It remains the case that our recall was low enough to drop our F-measure considerably. Will it be the case that a system like ours can scale to practical performance levels nonetheless? Four factors suggest that it can.

The first is that there is considerable redundancy in the data; although we have not quantified it for this data set, we note that the same event is often

	Tasks 1 and 3					Task 2		
Event class	GS	answer	R	P	F	R	P	F
Localization	174 (33)	41 (33)	18.97	80.49	30.70	16.67	69.05	26.85
Binding	347 (62)	152 (62)	17.87	40.79	24.85	17.48	40.13	24.35
Gene expression	722 (290)	344 (290)	40.17	84.30	54.41	40.17	84.30	54.41
Transcription	137 (28)	31 (28)	20.44	90.32	33.33	20.44	90.32	33.33
Protein catabolism	14 (4)	6 (4)	28.57	66.67	40.00	28.57	66.67	40.00
Phosphorylation	135 (47)	48 (47)	34.81	97.92	51.37	32.37	84.91	46.88
EVENT TOTAL	1529 (464)	622 (464)	30.35	74.60	43.14	29.77	72.77	42.26
Regulation	291 (11)	31 (11)	3.78	35.48	6.83	3.77	35.48	6.81
Positive regulation	983 (60)	129 (60)	6.10	46.51	10.79	6.08	46.51	10.75
Negative regulation	379 (18)	41 (18)	4.75	43.90	8.57	4.49	41.46	8.10
REGULATION TOTAL	1653 (89)	201 (89)	5.38	44.28	9.60	5.31	43.78	9.47
Negation	227 (6)	129 (6)	2.64	4.65	3.37			
Speculation	208 (25)	165 (25)	12.02	15.15	13.40			
MODIFICATION TOTAL	435 (31)	294 (31)	7.13	10.54	8.50			
ALL TOTAL	3182 (553)	823 (553)	17.38	67.19	27.62	17.07	65.74	27.10

Table 3: Updated results on test data for Tasks 1-3, with important bug fixes in the code base. See key above.

mentioned repeatedly, but for knowledge base building and other uses of the extracted information, it is only strictly necessary to recognize an event once (although multiple recognition of the same assertion may increase our confidence in its correctness).

The second is that there is often redundancy across the literature; the best-supported assertions will be reported as initial findings and then repeated as background information.

The third is that these recall results reflect an approach that made no use of syntactic analysis beyond handling coordination. There is often text present in the input that cannot be disregarded without either using wildcards, which generally decreased precision in our experiments and which we generally eschewed, or making use of syntactic information to isolate phrasal heads. Syntactic analysis, particularly when combined with analysis of predicate-argument structure, has recently been shown to be an effective tool in biomedical information extraction (Miyao et al., 2009). There is broad need for this—for example, of the thirty localization events in the training data whose trigger word was *translocation*, a full eighteen had intervening textual material that made it impossible for simple patterns like $translocation of [Theme]$ or $[ToLoc] translocation$ to match.

Finally, our recall numbers reflect a very short development cycle, with as few as four patterns written for many event types. A less time-constrained pattern-writing effort would almost certainly result in increased recall.

Acknowledgments

We gratefully acknowledge Mike Bada's help in loading the Sequence Ontology into Protégé.

This work was supported by NIH grants R01LM009254, R01GM083649, and R01LM008111 to Lawrence Hunter and T15LM009451 to Philip Ogren.

References

Alias-i. 2008. LingPipe 3.1.2.

Ekaterina Buyko, Joachim Wermter, Michael Poprat, and Udo Hahn. 2006. Automatically mapping an NLP core engine to the biology domain. In *Proceedings of the ISMB 2006 joint BioLINK/Bio-Ontologies meeting*.

K. B. Cohen, L. Tanabe, S. Kinoshita, and L. Hunter. 2004. A resource for constructing customized test suites for molecular biology entity identification systems. *BioLINK 2004*, pages 1–8.

K. Bretonnel Cohen, Martha Palmer, and Lawrence Hunter. 2008. Nominalization and alternations in biomedical language. *PLoS ONE*, 3(9).

D. Ferrucci and A. Lally. 2004. Building an example application with the unstructured information management architecture. *IBM Systems Journal*, 43(3):455–475, July.

Lawrence Hunter, Zhiyong Lu, James Firby, William A. Baumgartner Jr., Helen L. Johnson, Philip V. Ogren, and K. Bretonnel Cohen. 2008. OpenDMAP: An open-source, ontology-driven concept analysis engine, with applications to capturing knowledge regarding protein transport, protein interactions and cell-specific gene expression. *BMC Bioinformatics*, 9(78).

IBM. 2009. UIMA Java framework. http://uima-framework.sourceforge.net/.

Jin-Dong Kim, Tomoko Ohta, Sampo Pyysalo, Yoshinobu Kano, and Jun'ichi Tsujii. 2009. Overview of BioNLP'09 shared task on event extraction. In *Proceedings of Natural Language Processing in Biomedicine (BioNLP) NAACL 2009 Workshop*. To appear.

Yusuke Miyao, Kenji Sagae, Rune Saetre, Takuya Matsuzaki, and Jun'ichi Tsujii. 2009. Evaluating contributions of natural language parsers to protein-protein interaction extraction. *Bioinformatics*, 25(3):394–400.

Karin Verspoor, K. Bretonnel Cohen, and Lawrence Hunter. In press. The textual characteristics of traditional and Open Access scientific journals are similar. *BMC Bioinformatics*.

A memory–based learning approach to event extraction in biomedical texts

Roser Morante, Vincent Van Asch, Walter Daelemans
CNTS - Language Technology Group
University of Antwerp
Prinsstraat 13
B-2000 Antwerpen, Belgium
{Roser.Morante,Walter.Daelemans,Vincent.VanAsch}@ua.ac.be

Abstract

In this paper we describe the memory-based machine learning system that we submitted to the BioNLP Shared Task on Event Extraction. We modeled the event extraction task using an approach that has been previously applied to other natural language processing tasks like semantic role labeling or negation scope finding. The results obtained by our system (30.58 F-score in Task 1 and 29.27 in Task 2) suggest that the approach and the system need further adaptation to the complexity involved in extracting biomedical events.

1 Introduction

In this paper we describe the memory-based machine learning system that we submitted to the BioNLP shared task on event extraction[1]. The system operates in three phases. In the first phase, event triggers and entities other than proteins are detected. In the second phase, event participants and arguments are identified. In the third phase, postprocessing heuristics select the best frame for each event.

Memory-based language processing (Daelemans and van den Bosch, 2005) is based on the idea that NLP problems can be solved by reuse of solved examples of the problem stored in memory. Given a new problem, the most similar examples are retrieved, and a solution is extrapolated from them. As language processing tasks typically involve many subregularities and (pockets of) exceptions, it has been argued that memory-based learning is at an advantage in solving these highly disjunctive learning problems compared to more eager learning that abstract from the examples, as the latter eliminates not only noise but also potentially useful exceptions (Daelemans et al., 1999).

The BioNLP Shared Task 2009 takes a linguistically-motivated approach, which is reflected in the properties of the shared task definition: rich semantics, a text-bound approach, and decomposition of linguistic phenomena. Memory-based algorithms have been successfully applied in language processing to a wide range of linguistic tasks, from phonology to semantic analysis. Our goal was to investigate the performance of a memory–based approach to the event extraction task, using only the information available in the training corpus and modelling the task applying an approach similar to the one that has been applied to tasks like semantic role labeling (Morante et al., 2008) or negation scope detection (Morante and Daelemans, 2009).

In Section 2 we briefly describe the task. Section 3 reviews some related work. Section 4 presents the system, and Section 5 the results. Finally, some conclusions are put forward in Section 6.

2 Task description

The BioNLP Shared Task 2009 on event extraction consists of recognising bio-molecular events in biomedical texts, focusing on molecular events involving proteins and genes. An event is defined as a relation that holds between multiple entities that fulfil different roles. Events can participate in one type

[1]Web page: `http://www-tsujii.is.s.u-tokyo.ac.jp/GENIA/SharedTask/index.html`

of events: regulation events.

The task is divided into the three subtasks listed below. We participated in subtasks 1 and 2.

- Task 1: event detection and characterization. This task involves event trigger detection, event typing, and event participant recognition.
- Task 2: event argument recognition. Recognition of entities other than proteins and the assignment of these entities as event arguments.
- Task 3: recognition of negations and speculations.

The task did not include a named entity recognition subtask. A gold standard set of named entity annotations for proteins was provided by the organisation. A dataset based on the publicly available portion of the GENIA (Collier et al., 1999) corpus annotated with events (Kim et al., 2008) and of the BioInfer (Pyysalo et al., 2007) corpus was provided for training, and held-out parts of the same corpora were provided for development and testing.

The inter-annotator agreement reported for the Genia Event corpus is 56% strict match[2], which means that the event type is the same, the clue expressions are overlapping and the themes are the same. This low inter-annotator agreement is an indicator of the complexity of the task. Similar low inter-annotator agreement rates (49.00 %) in identification of events have been reported by Sasaki et al. (2008).

3 Related work

In recent years, research on text mining in the biomedical domain has experienced substantial progress, as shown in reviews of work done in this field (Krallinger and Valencia, 2005; Ananiadou and McNaught, 2006; Krallinger et al., 2008b). Some corpora have been annotated with event level information of different types: PropBank-style frames (Wattarujeekrit et al., 2004; Chou et al., 2006), frame independent roles (Kim et al., 2008), and specific roles for certain event types (Sasaki et al., 2008). The focus on extraction of event frames using machine learning techniques is relatively new because there were no corpora available.

Most work focuses on extracting biological relations from corpora, which consists of finding associations between entities within a text phrase. For example, Bundschus et al. (2008) develop a Conditional Random Fields (CRF) system to identify relations between genes and diseases from a set of GeneRIF (Gene Reference Into Function) phrases. A shared task was organised in the framework of the Language Learning in Logic Workshop 2005 devoted to the extraction of relations from biomedical texts (Nédellec, 2005). Extracting protein-protein interactions has also produced a lot of research, and has been the focus of the BioCreative II competition (Krallinger et al., 2008a).

As for event extraction, Yakushiji et al. (2001) present work on event extraction based on full-parsing and a large-scale, general-purpose grammar. They implement an Argument Structure Extractor. The parser is used to convert sentences that describe the same event into an argument structure for this event. The argument structure contains arguments such as semantic subject and object. Information extraction itself is performed using pattern matching on the argument structure. The system extracts 23 % of the argument structures uniquely, and 24% with ambiguity. Sasaki et al. (2008) present a supervised machine learning system that extracts event frames from a corpus in which the biological process *E. coli gene regulation* was linguistically annotated by domain experts. The frames being extracted specify all potential arguments of gene regulation events. Arguments are assigned domain-independent roles (Agent, Theme, Location) and domain-dependent roles (Condition, Manner). Their system works in three steps: (i) CRF-based named entity recognition to assign named entities to word sequences; (ii) CRF-based semantic role labeling to assign semantic roles to word sequences with named entity labels; (iii) Comparison of word sequences with event patterns derived from the corpus. The system achieves 50% recall and 20% precision.

We are not aware of work that has been carried out on the data set of the BioNLP Shared Task 2009 before the task took place.

[2]We did not find inter-annotator agreement measures in the paper that describes the corpus (Kim et al., 2008), but in `www-tsujii.is.s.u-tokyo.ac.jp/T-FaNT/T-FaNT.files/Slides/Kim.pdf`.

4 System description

We developed a supervised machine learning system. The system operates in three phases. In the first phase, event triggers and entities other than proteins are detected. In the second phase, event participants and arguments are identified. In the third phase, postprocessing heuristics select the best frame for each event. Parameterisation of the classifiers used in Phases 1 and 2 was performed by experimenting with sets of parameters on the development set. We experimented with manually selected parameters and with parameters selected by a genetic algorithm, but the parameters found by the genetic algorithm did not yield better results than the manually selected parameters

As a first step, we preprocess the corpora with the GDep dependency parser (Sagae and Tsujii, 2007) so that we can use part-of-speech tags and syntactic information as features for the machine learner. GDep is a a dependency parser for biomedical text trained on the Tsujii Lab's GENIA treebank. The dependency parser predicts for every word the part-of-speech tag, the lemma, the syntactic head, and the dependency relation. In addition to these regular dependency tags it also provides information about the IOB-style chunks and named entities. The classifiers use the output of GDep in addition to some frequency measures as features.

We represent the data into a columns format, following the standard format of the CoNLL Shared Task 2006 (Buchholz and Marsi, 2006), in which sentences are separated by a blank line and fields are separated by a single tab character. A sentence consists of tokens, each one starting on a new line.

4.1 Phase 1: Entity Detection

In the first phase, a memory based classifier predicts for every word in the corpus whether it is an entity or not and the type of entity. In this setting, *entity* refers to what in the shared task definition are events and entities other than proteins. Classes are defined in the IOB-style[3] in order to find entities that span over multiple words. Figure 1 shows a simplified version of a sentence in which *high level* is a Positive Regulation event that spans over multiple tokens and *proenkephalin* is a Protein. The Protein class does not need to be predicted by the classifier because this information is provided by the Task organisers. The classes predicted are: O, {B,I}-Entity, {B,I}-Binding, {B,I}-Gene Expression, {B,I}-Localization, {B,I}-Negative Regulation, {B,I}-Positive Regulation, {B,I}-Phosphorylation, {B,I}-Protein Catabolism, {B,I}-Transcription.

Token	Class	Token	Class
Upon	O	which	O
activation	O	correlate	O
,	O	with	O
T	O	high	B-Positive_regulation
lymphocyte	O	level	I-Positive_regulation
accumulate	O	of	O
high	O	proenkephalin	B-Protein
level	O	mRNA	O
of	O	in	O
the	O	the	O
neuropeptide	O	cell	O
enkephalin	O	.	O

Figure 1: Instance representation for the entity detection classifier.

We use the IB1 memory–based classifier as implemented in TiMBL (version 6.1.2) (Daelemans et al., 2007), a supervised inductive algorithm for learning classification tasks based on the k-nearest neighbor classification rule (Cover and Hart, 1967). The memory-based learning algorithm was parameterised in this case by using modified value difference as the similarity metric, gain ratio for feature weighting, using 7 k-nearest neighbors, and weighting the class vote of neighbors as a function of their inverse linear distance. For training we did not use the entire set of instances from the training data. We downsampled the instances keeping 5 negative instances (class label O) for every positive instance. Instances to be kept were randomly selected. The features used by this classifier are the following:

- About the token in focus: word, chunk tag, named entity tag as provided by the dependency parser, and, for every entity type, a number indicating how many times the focus word triggered this type of entity in the training corpus.
- About the context of the token in focus: lemmas ranging from the lemma at position -4 until the lemma at position +3 (relative to the focus word); part-of-speech ranging from position -1 until position +1; chunk ranging from position -1 until position +1 relative to the focus word; the chunk be-

[3]*I* stands for 'inside', *B* for 'beginning', and *O* for 'outside'.

fore the chunk to which the focus word belongs; a boolean indicating if a word is a protein or not for the words ranging from position -2 until position +3.

Class label	Precision	Recall	F-score
B-Gene_expression	59.32	60.23	59.77
B-Regulation	30.41	33.58	31.91
B-Entity	40.21	41.49	40.84
B-Positive_regulation	41.16	46.25	43.56
B-Binding	57.76	53.14	55.36
B-Negative_regulation	42.94	48.67	45.63
I-Negative_regulation	7.69	3.33	4.65
I-Positive_regulation	14.29	13.24	13.74
B-Phosphorylation	75.68	71.80	73.68
I-Regulation	14.29	10.00	11.77
B-Transcription	48.78	59.70	53.69
I-Entity	20.00	16.13	17.86
B-Localization	75.00	60.00	66.67
B-Protein_catabolism	73.08	100.00	84.44
O	97.66	97.62	97.64

Table 1: Results of the entity detection classifier. Entities that are not in the table have a precision and recall of 0.

Table 1 shows the results[4] of this first step. All class labels with a precision and recall of 0 are left out. The overall accuracy is 95.4%. This high accuracy is caused by the skewness of the data in the training corpus, which contains a higher proportion of instances with class label O. Instances with this class are correctly classified in the development test. B-Protein_catabolism and B-Phosphorylation get the highest scores. The reason why these classes get higher scores can be that the words that trigger these events are less diverse.

4.2 Phase 2: predicting the arguments and participants of events

In the second phase, another memory-based classifier predicts the participants and arguments of an event. Participants have the main role in the event and arguments are entities that further specify the event. In (1), for the event *phosphorylation* the system has to find that *STAT1, STAT3, STAT4, STAT5a,* and *STAT5b* are participants with the role *Theme* and that *tyrosine* is an argument with the role *Site*.

[4]In this section we provide results on development data because the gold test data have not been made available.

(1) IFN-alpha enhanced tyrosine phosphorylation of STAT1, STAT3, STAT4, STAT5a, and STAT5b.

We use the IB1 algorithm as implemented in TiMBL (version 6.1.2) (Daelemans et al., 2007). The classifier was parameterised by using gain ratio for feature weighting, overlap as distance metrics, 11 nearest neighbors for extrapolation, and normal majority voting for class voting weights.

For this classifier, instances represent combinations of an event with all the entities in a sentence, for as many events as there are in a sentence. Entities include entities and events. We use as input the output of the classifier in Phase 1, so only events and entities classified as such in Phase 1, and the gold proteins will be combined. Events can have participants and arguments in a sentence different that their sentence. We calculated that in the training corpus these cases account for 5.54% of the relations, and decided to restrict the combinations at the sentence level. For the sentence in (1) above, where *tyrosine, phosphorylation, STAT1, STAT3, STAT4, STAT5a*, and *STAT5b* are entities and of those only *phosphorylation* is an event, the instances would be produced by combining *phosphorylation* with the seven entities.

The features used by this classifier are the following:

- Of the event and of the combined entity: first word, last word, type, named entity provided by GDep, chain of lemmas, chain of part-of-speech (POS) tags, chain of chunk tags, dependency label of the first word, dependency label of the last word.
- Of the event context and of the combined entity context: word, lemma, POS, chunk, and GDep named entity of the five previous and next words.
- Of the context between event and combined entity: the chain of chunks in between, number of tokens in between, a binary feature indicating whether event is located before or after entity.
- Others: four features indicating the parental relation between the first and last words of the event and the first and last words of the entity. The values for this feature are: event_father, event_ancestor, entity_father, entity_ancestor, none. Five binary features indicating if the event accepts certain roles (Theme, Site, ToLoc, AtLoc, Cause).

Table 2 shows the results of this classifier per type of participant (Cause, Site, Theme) and type of argument (AtLoc, ToLoc). Arguments are very infrequent, and the participants are skewed towards the class Theme. Classes Site and Theme score high F1, and in both cases recall is higher than precision. The fact that the classifier overpredicts Sites and Themes will have a negative influence in the final scores of the full system. Further research will focus on improving precision.

Part/Arg	Total	Precision	Recall	F1
Cause	61	28.88	21.31	24.52
Site	20	54.83	85.00	66.66
Theme	683	55.50	72.32	62.80
AtLoc	1	25.00	100.00	40.00
ToLoc	4	75.00	75.00	75.00

Table 2: Results of finding the event participants and arguments.

Table 3 shows the results of finding the event participants and arguments per event type, expressed in terms of accuracy on the development corpus. Cause is easier to predict for Positive Regulation events, Site is the easiest class to predict, taking into account that AtLoc and ToLoc occur only 5 times in total, and Theme can be predicted successfully for Transcription and Gene Expression events, whereas it gets lower scores for Regulation, Binding, and Positive Regulation events.

Event	Arguments/Participants				
Type	Cause	Site	Theme	AtLoc	ToLoc
Binding	-	100.00	56.00	-	-
Gene Expr.	-	-	89.95	-	-
Localization	-	-	73.07	100.00	75.00
- Regulation	11.11	0.00	75.00	-	-
Phosphorylation	0.00	100.00	70.83	-	-
+ Regulation	27.77	90.90	56.77	-	-
Protein Catab.	-	-	60.00	-	-
Regulation	13.33	0.00	46.87	-	-
Transcription	-	-	94.44	-	-

Table 3: Results of finding the event participants and arguments per event type (accuracy).

Table 4 shows the results of finding the event participants that are Entity and Protein per type of event for events that are not regulations. Entity scores high in all cases, whereas Protein scores high for Transcription and Gene Expression events and low for Binding events.

Event	Arg./Part. Type	
Type	Entity	Protein
Binding	100.00	56.00
Gene Expr.	-	89.90
Localization	80.00	73.07
Phosphorylation	100.00	68.00
Protein Catab.	-	60.00
Transcription	-	94.44

Table 4: Results of finding the event participants and arguments that are Entity and Protein per event type (accuracy).

Table 5 shows the results of finding the participants and arguments of regulation events. In the case of regulation events, Entity is easier to classify with Positive Regulation events, and Protein with Negative Regulation events. In the cases in which events are participants of regulation events, Binding, Gene Expression and Phosphorylation are easier to classify with Positive Regulation events, Localization with Regulation events, Protein Catabolism with Negative Regulation events, and Transcription is easy to classify in all cases.

Arg./Part.	Event Type		
Type	Regulation	+ Regulation	-Regulation
Entity	0.00	90.90	0.00
Protein	17.85	38.88	45.45
Binding	-	75.00	66.66
Gene Expr.	66.66	90.47	75.00
Localization	100.00	80.00	75.00
Phosphorylation	0.00	44.44	0.00
Protein Catab.	0.00	40.00	100.00
Transcription	100.00	92.85	100.00

Table 5: Results of finding event arguments and participants for regulation events (accuracy).

From the results of the system in this phase we can extract some conclusions: data are skewed towards the Theme class; Themes are not equally predictable for the different types of events, they are better predictable for Gene Expression and Transcription; Proteins are more difficult to classify when they are Themes of regulation events; and Transcription and Localization events are easier to predict as Themes of regulation events, compared to the other types of events that are Themes of regulation events. This suggests that it could be worth experimenting with a classifier per entity type and with a classifier per role, instead of using the same classifier for all types of entities.

4.3 Phase 3: heuristics to select the best frame per event

Phases 1 and 2 aimed at identifying events and candidates to event participants. However, the purpose of the task is to extract full frames of events. For a sentence like the one in (1) above, the system has to extract the event frames in (2).

(2)
1. Phosphorylation (phosphorylation): Theme (STAT1) Site (tyrosine)
2. Phosphorylation (phosphorylation): Theme (STAT3) Site (tyrosine)
3. Phosphorylation (phosphorylation): Theme (STAT5a) Site (tyrosine)
4. Phosphorylation (phosphorylation): Theme (STAT4) Site (tyrosine)
5. Phosphorylation (phosphorylation): Theme (STAT5b) Site (tyrosine)

It is necessary to apply heuristics in order to build the event frames from the output of the second classifier, which for the sentence in (1) above should contain the predictions in (3).

(3)
1. phosphorylation STAT1 : Theme
2. phosphorylation STAT3 : Theme
3. phosphorylation STAT5a : Theme
4. phosphorylation STAT4 : Theme
5. phosphorylation STAT5b : Theme
6. phosphorylation tyrosine : Site

Thus, in the third phase, postprocessing heuristics determine which is the frame of each event.

4.3.1 Specific heuristics for each type of event

The system contains different rules for each of the 5 types of participants (Cause, Site, Theme, AtLoc, ToLoc). The text entities are the entities defined during Phase 2. An event is created for every text entity for which the system predicted at least one participant or argument. To illustrate this we can take a look at the predictions for the Gene Expression event in (4) where the identifiers starting by T refer to entities in the text. The prediction would results in the events listed in (5).

(4) Gene_expression=
Theme:T11=Theme:T12=Theme:T13

(5) E1 Gene_expression:T23 Theme:T11
E2 Gene_expression:T23 Theme:T12
E3 Gene_expression:T23 Theme:T13

Gene expression, Transcription, and Protein catabolism. These type of events have only a Theme. Therefore, an event frame is created for every Theme predicted for events that belong to these types.

Localization. A Localization event can have one Theme and 2 arguments: AtLoc and ToLoc. A Localization event with more than one predicted Theme will result in as many frames as predicted Themes. The arguments are passed on to every frame.

Binding. A Binding event can have multiple Themes and multiple Site arguments. If the system predicts more than one Theme for a Binding event, the heuristics first check if these Themes are in a coordination structure. Coordination checking consists of checking whether the word 'and' can be found between the Themes. Coordinated Themes will give rise to separate frames. Every participant and loose Theme is added to all created event lines. This case applies to the sentence in (6)

(6) When we analyzed the nature of STAT proteins capable of binding to IL-2Ralpha, pim-1, and IRF-1 GAS elements after cytokine stimulation, we observed IFN-alpha-induced binding of STAT1, STAT3, and STAT4, but not STAT5 to all of these elements.

The frames that should be created for this sentence listed in (7).

(7)
1. Binding (binding): Theme(STAT4) Theme2(IRF-1) Site2(GAS elements)
2. Binding (binding): Theme(STAT3) Theme2:(IL-2Ralpha) Site2(GAS elements)
3. Binding (binding): Theme(STAT3) Theme2(IRF-1) Site2(GAS elements)
4. Binding (binding): Theme(STAT4) Theme2(pim-1) Site2(GAS elements)
5. Binding (binding): Theme(STAT1) Theme2(IL-2Ralpha) Site2(GAS elements)

6. Binding (binding): Theme(STAT4) Theme2(IL-2Ralpha) Site2(GAS elements)
7. Binding (binding): Theme(IL-2Ralpha) Site(GAS elements)
8. Binding (binding): Theme(pim-1) Site(GAS elements)
9. Binding (binding): Theme(STAT1) Theme2(IRF-1) Site2(GAS elements)
10. Binding (binding): Theme(STAT3) Theme2(pim-1) Site2(GAS elements)
11. Binding (binding): Theme(IRF-1) Site(GAS elements)
12. Binding (binding): Theme(STAT1) Theme2(pim-1) Site2(GAS elements)

Phosphorylation. A Phosphorylation event can have one Theme and one Site. Multiple Themes for the same event will result in multiple frames. The Site argument will be added to every frame.

Regulation, Positive regulation, and Negative regulation. A Regulation event can have a Theme, a Cause, a Site, and a CSite. For Regulation events the system uses a different approach when creating new frames. It first checks which of the participants and arguments occurs the most frequent in a prediction and it creates as many separate frames as are needed to give every participant/argument its own frame. The remaining participants/arguments are added to the nearest frame. For this type of event a new frame can be created not only for multiple Themes but also for e.g. multiple Sites. The purpose of this strategy is to increase the recall of Regulation events.

4.3.2 Postprocessing

After translating predictions into frames some corrections are made.

1. Every Theme and Cause that is not a Protein is thrown away.
2. Every frame that has no Theme is provided with a default Theme. If no Protein is found before the focus word, the closest Protein after the word is taken as the default Theme.
3. Duplicates are removed.

5 Results

The official results of our system for Task 1 are presented in Table 6. The best F1 score are for Gene Expression and Protein Catabolism events. The lowest results are for all the types of regulation events and for Binding events. Binding events are more difficult to predict correctly because they can have more than one Theme.

	Total	Precision	Recall	F1
Binding	347	12.97	31.03	18.29
Gene Expr.	722	51.39	68.96	58.89
Localization	174	20.69	78.26	32.73
Phosphorylation	135	28.15	67.86	39.79
Protein Catab.	14	64.29	42.86	51.43
Transcription	137	24.82	41.46	31.05
Regulation	291	8.93	23.64	12.97
+Regulation	983	11.70	31.68	17.09
-Regulation	379	11.08	29.85	16.15
TOTAL	3182	22.50	47.70	30.58

Table 6: Official results of Task 1. Approximate Span Matching/Approximate Recursive Matching.

The official results of our system for Task 2 are presented in Table 7. Results are similar to the results of Task 1 because there are not many more arguments than participants. Recognising arguments was the additional goal of Task 2 in relation to Task 1.

	Total	Precision	Recall	F1
Binding	349	11.75	28.28	16.60
Gene Expr.	722	51.39	68.96	58.89
Localization	174	17.82	67.39	28.18
Phosphorylation	139	15.83	39.29	22.56
Protein Catab.	14	64.29	42.86	51.43
Transcription	137	24.82	41.46	31.05
Regulation	292	8.56	22.73	12.44
+Regulation	987	11.35	30.85	16.59
-Regulation	379	11.08	29.20	15.76
TOTAL	3193	21.52	45.77	29.27

Table 7: Official results of Task 2. Approximate Span Matching/Approximate Recursive Matching.

Results obtained on the development set are a little bit higher. For Task1 an overall F1 of 34.78 and for Task 2 33.54.

For most event types precision and recall are unbalanced, the system scores higher in recall. Further research should focus on increasing precision because the system is predicting false positives. It would be possible to add a step in order to filter out the false positives by comparing word sequences with event patterns derived from the corpus, which is an approach taken in the system by Sasaki et al. (2008) .

In the case of Binding events, both precision and recall are low. There are two explanations for this. In the first place, the first classifier misses almost half of the binding events. As an example, for the sentence in (8.1), the gold standard identifies as binding event the multiwords *binds as a homodimer* and *form heterodimers*, whereas the system identifies two binding events for the same sentence, *binds* and *homodimer*, none of which is correct because the correct one is the multiword unit. For the sentence in (8.2), the gold standard identifies as binding events *bind*, *form homo-*, and *heterodimers*, whereas the system identifies only *binds*.

(8) 1. The KBF1/p50 factor *binds as a homodimer* but can also *form heterodimers* with the products of other members of the same family, like the c-rel and v-rel (proto)oncogenes.

2. A mutant of KBF1/p50 (delta SP), unable to *bind* to DNA but able to *form homo-* or *heterodimers*, has been constructed.

From the sentence in (8.1) above the eight frames in (9) should be extracted, whereas the system extracts only the frames in (10), which are incorrect because the events have not been correctly identified.

(9)
1. Binding(binds as a homodimer) : Theme(KBF1)
2. Binding(binds as a homodimer) : Theme(p50)
3. Binding(form heterodimers) : Theme(KBF1) Theme2(c-rel)
4. Binding(form heterodimers) : Theme(p50) Theme2(v-rel)
5. Binding(form heterodimers) : Theme(p50) Theme2(c-rel)
6. Binding(form heterodimers) : Theme(KBF1) Theme2(v-rel)
7. Binding(bind) : Theme(p50)
8. Binding(bind) : Theme(KBF1)

(10)
1. Binding(binds) : Theme(v-rel)
2. Binding(homodimer) : Theme(c-rel)

The complexity of frame extraction of Binding events contrasts with the less complex extraction of frames for Gene Expression events, like the one in sentence (11), where *expression* has been identified correctly by the system as an event and the frame in (12) has been correctly extracted.

(11) Thus, c-Fos/c-Jun heterodimers might contribute to the repression of DRA gene *expression*.

(12) Gene Expression(expression) : Theme(DRA)

6 Conclusions

In this paper we presented a supervised machine learning system that extracts event frames from biomedical texts in three phases. The system participated in the BioNLP Shared Task 2009, achieving an F-score of 30.58 in Task 1, and 29.27 in Task 2. The frame extraction task was modeled applying the same approach that has been applied to tasks like semantic role labeling or negation scope detection, in order to check whether such an approach would be suitable for a frame extraction task. The results obtained for the present task do not compare to results obtained in the mentioned tasks, where state of the art F-scores are above 80.

Extracting biomedical event frames is more complex than labeling semantic roles because of several reasons. Semantic roles are mostly assigned to syntactic constituents, predicates have only one frame and all the arguments belong to the same frame. In contrast, in the biomedical domain one event can have several frames, each frame having different participants, the boundaries of which do not coincide with syntactic constituents.

The system presented here can be improved in several directions. Future research will concentrate on increasing precision in general, and precision and recall of binding events in particular. Analysing in depth the errors made by the system at each phase will allow us to find the weaker aspects of the system. From the results of the system in the second phase we could draw some conclusions: data are skewed towards the Theme class; Themes are not equally predictable for the different types of events; Proteins are more difficult to classify when they are Themes of regulation events; and Transcription and Localization events are easier to predict as Themes of regulation events, compared to the other types of events that are Themes of regulation events. We plan to experiment with a classifier per entity type and with a classifier per role, instead of using the same classifier for all types of entities. Additionally, the effects of the postprocessing rules in Phase 3 will be evaluated.

Acknowledgments

Our work was made possible through financial support from the University of Antwerp (GOA project BIOGRAPH). We are grateful to two anonymous reviewers for their valuable comments.

References

S. Ananiadou and J. McNaught. 2006. *Text Mining for Biology and Biomedicine*. Artech House Books, London.

S. Buchholz and E. Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proc. of the X CoNLL Shared Task*, New York. SIGNLL.

M. Bundschus, M. Dejori, M. Stetter, V. Tresp, and H-P Kriegel. 2008. Extraction of semantic biomedical relations from text using conditional random fields. *BMC Bioinformatics*, 9.

W.C. Chou, R.T.H. Tsai, Y-S. Su, W. Ku, T-Y Sung, and W-L Hsu. 2006. A semi-automatic method for annotating a biomedical proposition bank. In *Proc. of ACL Workshop on Frontiers in Linguistically Annotated Corpora 2006*, pages 5–12.

N. Collier, H.S. Park, N. Ogata, Y. Tateisi, C. Nobata, T. Sekimizu, H. Imai, and J. Tsujii. 1999. The GENIA project: corpus-based knowledge acquisition and information extraction from genome research papers. In *Proc. of EACL 1999*.

T. M. Cover and P. E. Hart. 1967. Nearest neighbor pattern classification. *Institute of Electrical and Electronics Engineers Transactions on Information Theory*, 13:21–27.

W. Daelemans and A. van den Bosch. 2005. *Memory-based language processing*. Cambridge University Press, Cambridge, UK.

W. Daelemans, A. Van den Bosch, and J. Zavrel. 1999. Forgetting exceptions is harmful in language learning. *Machine Learning, Special issue on Natural Language Learning*, 34:11–41.

W. Daelemans, J. Zavrel, K. Van der Sloot, and A. Van den Bosch. 2007. TiMBL: Tilburg memory based learner, version 6.1, reference guide. Technical Report Series 07-07, ILK, Tilburg, The Netherlands.

J.D. Kim, T. Ohta, and J. Tsujii. 2008. Corpus annotation for mining biomedical events from literature. *BMC Bioinformatics*, 9:10.

M. Krallinger and A. Valencia. 2005. Text-mining and information-retrieval services for molecular biology. *Genome Biology*, 6:224.

M. Krallinger, F. Leitner, C. Rodriguez-Penagos, and A. Valencia. 2008a. Overview of the protein–protein interaction annotation extraction task of BioCreative II. *Genome Biology*, 9(Suppl 2):S4.

M. Krallinger, A. Valencia, and L. Hirschman. 2008b. Linking genes to literature: text mining, information extraction, and retrieval applications for biology. *Genome Biology*, 9(Suppl 2):S8.

R. Morante and W. Daelemans. 2009. A metalearning approach to processing the scope of negation. In *Proceedings of CoNLL 2009*, Boulder, Colorado.

R. Morante, W. Daelemans, and V. Van Asch. 2008. A combined memory-based semantic role labeler of English. In *Proc. of the CoNLL 2008*, pages 208–212, Manchester, UK.

C. Nédellec. 2005. Learning language in logic – genic interaction extraction challenge. In *Proc. of Learning Language in Logic Workshop 2005*, pages 31–37, Bonn.

S. Pyysalo, F. Ginter, J. Heimonen, J. Björne, J. Boberg, J. Järvinen, and T. Salakoski. 2007. BioInfer: a corpus for information extraction in the biomedical domain. *BMC Bioinformatics*, 8(50).

K. Sagae and J. Tsujii. 2007. Dependency parsing and domain adaptation with lr models and parser ensembles. In *Proc. of CoNLL 2007 Shared Task, EMNLP-CoNLL*, pages 82–94, Prague. ACL.

Y. Sasaki, P. Thompson, P. Cotter, J. McNaught, and S. Ananiadou. 2008. Event frame extraction based on a gene regulation corpus. In *Proc. of Coling 2008*, pages 761–768.

T. Wattarujeekrit, P.K. Shah, and N. Collier. 2004. PASBio: predicate-argument structures for event extraction in molecular biology. *BMC Bioinformatics*, 5:155.

A. Yakushiji, Y. Tateisi, Y. Miyao, and J. Tsujii. 2001. Event extraction from biomedical papers using a full parser. In *Pac Symp Biocomput*.

Extraction of biomedical events using case-based reasoning

Mariana L. Neves
Biocomputing Unit
Centro Nacional de Biotecnología - CSIC
C/ Darwin 3, Campus de Cantoblanco,
28049, Madrid, Spain
mlara@cnb.csic.es

José M. Carazo
Biocomputing Unit
Centro Nacional de Biotecnología - CSIC
C/ Darwin 3, Campus de Cantoblanco,
28049, Madrid, Spain
carazo@cnb.csic.es

Alberto Pascual-Montano
Departamento de Arquitectura de Computadores
Universidad Complutense de Madrid, Facultad de
Ciencias Físicas
28040, Madrid, Spain
pascual@fis.ucm.es

Abstract

The BioNLP´09 Shared Task on Event Extraction presented an evaluation on the extraction of biological events related to genes/proteins from the literature. We propose a system that uses the case-based reasoning (CBR) machine learning approach for the extraction of the entities (events, sites and location). The mapping of the proteins in the texts to the previously extracted entities is carried out by some simple manually developed rules for each of the arguments under consideration (cause, theme, site or location). We have achieved an f-measure of 24.15 and 21.15 for Task 1 and 2, respectively.

1 Introduction

The increasing amount of biological data generated by the high throughput experiments has lead to a great demand of computational tools to process and interpret such amount of information. The protein-protein interactions, as well as molecular events related to one entity only, are key issues as they take part in many biological processes, and many efforts have been dedicate to this matter. For example, databases are available for the storage of such interaction pairs, such as the Molecular INTeraction Database (Chatr-aryamontri et al., 2007) and IntAct (Kerrien et al., 2007).

In the field of text mining solutions, many efforts have been made. For example, the BioCreative II protein-protein interaction (PPI) task (Krallinger, Leitner, Rodriguez-Penagos, & Valencia, 2008) consists of four sub-tasks, including the extraction of the protein interaction pairs in full-text documents, achieving an f-measure of up to 0.30. The initiative of annotation of both Genia corpus (J. D. Kim, Ohta, & Tsujii, 2008) and BioInfer (Pyysalo et al., 2007) is another good example.

The BioNLP´09 Shared Task on Event Extraction (J.-D. Kim, Ohta, Pyysalo, Kano, & Tsujii, 2009) proposes a comparative evaluation for the extraction of biological events related to one or more gene/protein and even other types of entities related to the localization of the referred event in the cell. The types of events that have been considered in the shared task were localization, binding, gene expression, transcription, protein catabolism, phosphorylation, regulation, positive regulation and negative regulation. A corpus that consisted of 800, 150 and 260 PubMed documents (title and abstract text only) was made available for the training, development test and testing datasets, respectively. For all documents, the proteins that took part in the events were provided.

The shared task organization proposed three tasks. Task 1 (Event detection and characterization) required the participants to extract the events from the text and map them to its respec-

Proceedings of the Workshop on BioNLP: Shared Task, pages 68–76,
Boulder, Colorado, June 2009. ©2009 Association for Computational Linguistics

tive theme(s), as an event may be associated to one or more themes, e.g. binding. Also, some events may have only a gene/protein as theme, e.g. protein catabolism, while some other may be also associated to another event, e.g. regulation events. Task 2 (Event argument recognition) asked the participants to provide the many arguments that may be related to the extracted event, such as its cause, that may be an annotated or one of the previously extracted events. Other arguments include site and localization, which should be first extracted from the texts by the system, as they do not come annotated in the documents. Task 3 (Recognition of negation and speculations) evaluates the presence of negations and speculation related to the previously extracted events.

Our group has participated in this shared task with a system implemented with the case-based reasoning (CBR) machine learning technique as well as some manual rules. We have presented results for tasks 1 and 2 exclusively. The system described here is part of the Moara project[1] and was developed in Java programming language and use MySQL database.

2 Methods

Case-based reasoning (CBR) (Aamodt & Plaza, 1994) is the machine learning method that was used for extracting the terms and events here proposed and consists of first learning cases from the training documents, by means of saving them in a base of case, and further retrieving a case the most similar to a given problem during the testing step, from which will be given the final solution, hereafter called "case-solution". One of the advantages of the CBR algorithm is the possibility of getting an explanation of why to a given token has been attributed a certain category, by means of checking the features that compose the case-solution. Additionally, and due to the complexity of the tasks, a rule-based post-processing step was built in order to map the previously extracted terms and events among themselves.

2.1 Retaining the cases

In this first step, documents of the training dataset are tokenized according to spaces and punctuations. The resulting tokens are represented in the CBR approach as cases composed of some predefined features that take into account the morphology and grammatical function of the tokens in the text as well as specific features related to the problem under consideration. The resulting cases are then stored in a base of case to be further retrieved (Figure 1).

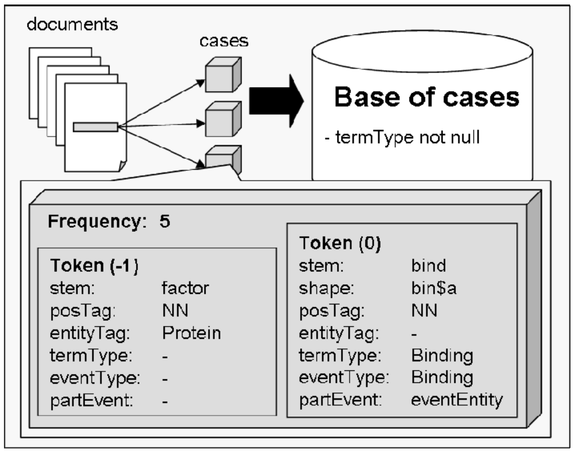


Figure 1: Training step in which cases are represented by some pre-defined features and further saved to a base.

Regarding the features that compose a case, these were the ones that were considered during the training and development phases: the token itself (token); the token in lower case (lowercase); the stem of the token (stem); the shape of the token (shape); the part-of-speech tag (posTag); the chunk tag (chunkTag); a biomedical entity tag (entityTag); the type of the term (termType); the type of the event (eventType); and the part of the term in the event (eventPart). The stem of a token was extracted using an available Java implementation[2] of the Porter algorithm (Porter, 1980), while the part-of-speech, chunk and bio-entity tags were taken from the GENIA Tagger (Tsuruoka et al., 2005).

The shape of a token is given by a set of characters that represent its morphology: "a" for lower case letters, "A" for upper case letters, "1" for numbers, "g" for Greek letters, "p" for stop-

[1] http://moara.dacya.ucm.es

[2] http://www.tartarus.org/~martin/PorterStemmer

words[3], “$” for identifying 3-letters prefixes or suffixes or any other symbol represented by itself. Here are some few example for the shape feature: “Dorsal” would be represented by “Aa”, “Bmp4” by “Aa1”, “the” by “p”, “cGKI(alpha)” by “aAAA(g)”, “patterning” by “pat$a” (‘$’ symbol separating the 3-letters prefix) and “activity” by “a$vity” (‘$’ symbol separating the 4-letters suffix). No repetition is allowed in the case of the “a” symbol for the lower case letters.

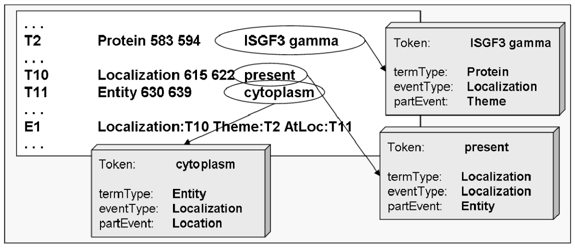


Figure 2: Example of the termType, eventType and partEvent features.

The last three features listed above are specific to the event detection task and were extracted from the annotation files (.a1 and .a2) that are part of the corpus. The termType feature is used to identify the type of the term in the event problem, and it is extracted from the term lines of both annotation files .a1 and .a2, i.e. the ones which the identifiers starts with a “T”. The eventType features represent the event itself and it is extracted from the event lines of .a2 annotation file, i.e. the ones that starts with an “E”. Finally, eventPart represents the token according to its role, i.e. entity, theme, cause, site and location. The termType, eventType and eventPart features are the hereafter called “feature-problem”, the features that are unknown to the system in the testing phase and which values are to be given by the case-solution. Figure 2 illustrate one example of these features for an extract of the annotation of the document “1315834” from the training dataset.

Usually, one case corresponds for each token of the documents in the training dataset. However, more than one case may be created from a token, as well as none at all, depending on the predefined features. For example, some tokens may derive in more than one case due to the shape feature, as for example, “patterning” (“pat$a”, “a$ing”, “a”). Also, according to the retaining strategy, some tokens may be associated to no case at all, for example, by restricting the value of a determined feature as the retaining strategy. In order to reduce the number of retained cases, and consequently reduce the further retrieving time, only those tokens related to an event are retained, i.e., tokens with not null value for the termType feature.

The text of a document may be read in the forward or backward direction during the training step, and even combining both of them (Neves, Chagoyen, Carazo, & Pascual-Montano, 2008). Here, we have considered the forward direction exclusively. Also, another important point is the window of tokens under consideration when setting the features of a case, if taking into account only the token itself or also the surrounding tokens, the ones which come before or after it. Here we consider a window of (-1,0), i.e., for each token, we get the feature of the token itself and of the preceding one, exclusively.

Features / Tokens	Training		Testing	
	-1	0	-1	0
stem	✓	✓	✓	✓
shape		✓		✓
posTag	✓	✓	✓	✓
chunkTag				
entityTag	✓	✓	✓	✓
termType	✓	✓	✓	✓
eventType	✓	✓	✓	
partEvent	✓	✓	✓	

Table 1: Selected features in the training and testing steps for the tokens “0” and “-1”. The last three features are the ones to be inferred.

Many experiments have been carried out in order to choose the best set of features (Table 1). The higher the number of features under consideration, the greater is the number of cases to be retained and the higher is the time needed to search for the case-solution. He relies therefore the importance of choosing a small an efficient set of features. For this reason, the shape features has not been considered for the preceding token (-1) in order to reduce the number of cases, as this shape usually result in more than one case per token. The termType feature is at the same time known and unknown in the testing step. It is know for the protein terms but is un-

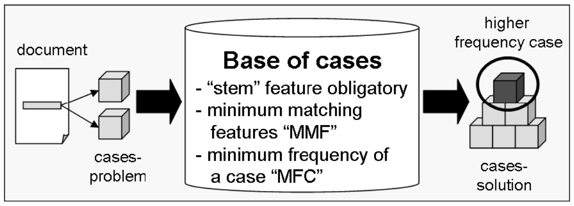
3 http://www.dcs.gla.ac.uk/idom/ir_resources/linguistic_utils/

known for the remaining entities (events, sites and locations).

By considering these features for the 800 documents in the training set, about 26,788 unique cases were generated. It should be noted that no repetition of cases with the same values for the features are allowed, instead a field for the frequency of the case is incremented to keep track of the number of times that it has appeared during the training phase. The frequency range goes from 1 (more than 22,000 cases) to 238 (one case only).

2.2 Retrieving a case

When a new document is presented to the system, it is first read in the forward direction and tokenized according to space and punctuation and the resulting tokens are mapped to cases of features, exactly as discussed in the retaining step. The only difference here is the set of feature (cf. Table 1), as some of them are unknown to the system and are the ones to be inferred from the cases retained during the training step.


document
cases-problem
Base of cases
- "stem" feature obligatory
- minimum matching features "MMF"
- minimum frequency of a case "MFC"
higher frequency case
cases-solution
Figure 3: Retrieval procedure to choose the most case-solution with higher frequency and based on MMF and MFC parameters.

For each token, the system first creates a case (hereafter called "case-problem") based on the testing features and proceeds to search the base of cases for the case-solution the most similar to this case-problem (Figure 3). It should be noted that a token may have more than one case-problem, depending of the values of the shape feature. The best case-solution among the ones found by the system will be the one with the higher frequency. The system always tries to find a case-solution with the higher number of features that have exactly the same value of the case-problem's respective features. The stem is the only mandatory feature which value must be always matched between the case-problem and the case-solution. The value of the two features-problem (eventType and partEvent) will be given by the values of the case-solution's respective features. If no case solution is found, the token is considered of not being related to the event domain in none of its parts (entity, theme, cause, etc.).

Two parameters have been taken into consideration in the retaining strategy: the minimum matching feature (MMF) and the minimum frequency of the case (MFC). The first one set the minimum features that should be matched between the case-problem and the case-solution, as the higher the number of equal features between theses cases, the more precise is the decision inferred from the case-solution.

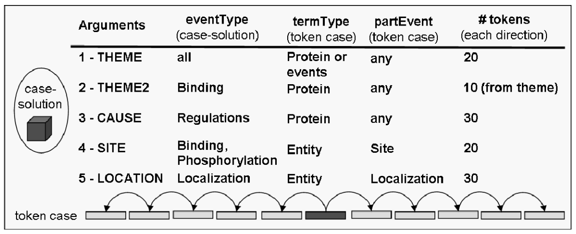
On the other hand, the MFC parameter restricts the cases that are to be considered by the search strategy, the ones with frequency higher than the value specified by this parameter. The higher the minimum frequency asked for a case, the lower is the number of cases under consideration and the lower is the time for obtaining the case-solution. From the 26,788 cases we have retained during the training phase, about 22,389 of them appeared just once and would not be considered by the searching procedure if the MFC parameter was set to 2, for example, therefore reducing the searching time.

Experiments have been carried out in order to decide the values for both parameters and it resulted that a better performance is achieved (cf. 3) by setting the MFC to a value higher than 1. On the other hand, experiments have shown that the recall may decrease considerably when restricting the MMF parameter.

By repeating this procedure for all the tokens of the document, the latter may be then considered as being tagged with the event entities. However, in order to construct the output file required by the shared task organization, some manual rules have been created in order to map the events mapped to its respective arguments, as described in the next section.

2.3 Post-processing rules

For the tasks 1 and 2, the participants were asked to output the events present in the provided texts along with their respective arguments. The events have been already extracted in the previous step; the tokens that were tagged as "Entity" for the "partEvent" feature (cf. Figure 2), hereafter called “event-entity”. This entity is the start point from which to search for the arguments which are incrementally extracted from the text in the following order: theme, theme 2, cause, site and location. Figure 4 resumes the rules for each of the arguments.



Arguments	eventType (case-solution)	termType (token case)	partEvent (token case)	# tokens (each direction)
1 - THEME	all	Protein or events	any	20
2 - THEME2	Binding	Protein	any	10 (from theme)
3 - CAUSE	Regulations	Protein	any	30
4 - SITE	Binding, Phosphorylation	Entity	Site	20
5 - LOCATION	Localization	Entity	Localization	30

case-solution
token case

Figure 4: Resume of the post-processing rules for each type of argument.

Themes: The theme-candidates for an event-entity are the annotated proteins (.a1 file) as well as the events themselves, in the case of the regulation, positive regulation and negative regulation events. The first step is then to try to map each event to its theme and in case that no theme is found, the event is not considered anymore by the system and it is not printed to the output file.

The theme searching strategy starts from the event-entity and consists of reading the text in both directions alternatively, one token in the forward direction followed by one token in the backward direction until a theme-candidate is found (Figure 5). The system halts if the end of the sentence is found or if the specified number of tokens in each direction is reached, 20 for the theme. By analyzing some of the false negatives returned from the experiments with the development dataset, we have learned that few events are associated to themes present in a different sentence and although aware these cases, we have decided to restrict the searching to the sentence boundaries in order to avoid a high number of false positives.

In the case of a second theme, allowed for binding events only, a similar searching strategy is carried out, except that here the system reads up of 10 tokens in each direction, starting from the theme entity previously extracted.

Cause: The cause-candidates are also the annotated proteins and, starting from the event-entity, a similar search is carried out, restricted up to 30 tokens in each direction and to the boundaries of the same sentence. This procedure is carried out for the regulation, positive regulation and negative regulation events only and the only extra restriction is that the candidate should not be the protein already assigned as theme. If no candidate is found, the system considers that there is no cause associated to the event under consideration.

Site and Location: Here the candidates are the tokens tagged with the values of “Entity” for the termType feature, and “Site” and “Location” for the partEvent feature, respectively. The search for the site is carried out for the binding and phosphorylation events and the location search for the localization event only. The procedure is restricted to the sentence boundaries and up to 20 and 30 tokens, respectively, starting from the event-entity. Once again, if not candidate is found, the system consider that there is no site or location associated to the event under consideration.

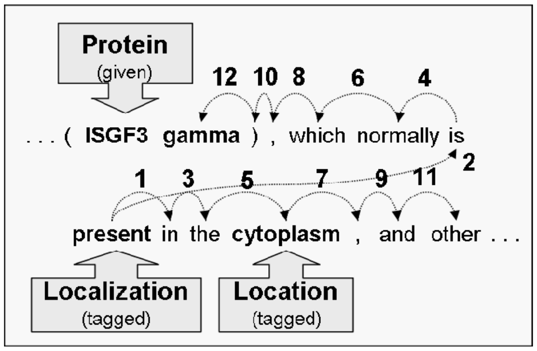

Protein
(given)
12 10 8 6 4
. . . (ISGF3 gamma) , which normally is
2
1 3 5 7 9 11
present in the cytoplasm , and other . . .
Localization
(tagged)
Location
(tagged)

Figure 5: Contribution of each class of error to the 275 false positives analyzed here.

3 Results

This section presents the results of the experiments carried out with the development and the blind test datasets as well as an analysis of the false negatives and false positives. Results here will be presented for tasks 1 and 2 in terms of precision, recall and f-measure.

Experiments have been carried out with the development dataset in order to decide the best value of the MMF and MFC parameters (cf. 2.2). Figure 6 shows the variation of the F-measure according to both parameters for the values of 1, 3, 4, 5, 6, 7 and 8 for MMF; and 1, 2, 5, 10, 15, 20 and 50 for MFC.

Usually, recall is higher for a low value of MFC, as the searching for the case-solution is carried out over a greater number of cases and the possibility of finding a good case-problem is higher. On the other hand, precision increases when few cases are under considered by the search strategy, as fewer decisions are taken and the cases-solution have usually a high frequency, avoiding decision based on "weak" cases of frequency 1, for example.

Figure 6 shows that the best value for MFC ranges from 2 to 20 and for MMF from 5 to 7 and the best f-measure result is found for the values of 2 and 6 for these parameters (f2m6), respectively. As these experiments have been carried out after the deadline of the test dataset, the run that was submitted as the final solution was the one with the values of 2 and 1 for the MFC and MMF parameters (f2m1), respectively. Table 3 and 4 resumes the results obtained for the test dataset with the configuration that was submitted (f2m1), and the best one (f2m6) after accomplishing the experiments above described. Results have slightly improved by only trying to choose the best values for the parameters here considered.

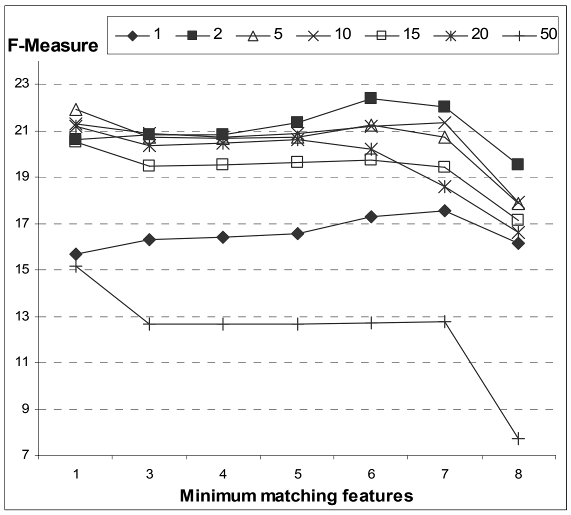


Figure 6: F-Measure for the development dataset in terms of the MFC (curves) and the MMF (x-axis).

An automatic analysis of the false positives and false negatives has been performed for the development dataset and for the results obtained with the final submission (f2m1), a total of 2502 false positives and 1300 false negatives. We have found out that the mistakes are related mainly to the retrieving of the case-solution and to the mapping of an event to its arguments. The mistakes have been classified in seven groups described below and figures 7 and 8 show the percent contribution of each class for the false positives and false negatives, respectively.

Events composed of more than one token (1): this mistake happens when the system is able to find the event with its correct type and arguments but with only part of its tokens, such as "regulation" instead of "up-regulation" and "reduced" or "levels" instead of "reduced levels", both in document 10411003. This is mainly due to our tokenization strategy of separating the tokens according to all punctuation and symbols (including hyphens) and also due to the evaluation method that seems not consider alternatives to the text of an event. This mistake always results in one false positive and one false negative.

tasks / results		**recall**	**precision**	**f-measure**
task 1	**(f2m1)**	28.63	20.88	24.15
	(f2m6)	27.18	23.92	25.45
task 2	**(f2m1)**	25.02	18.32	21.15
	(f2m6)	24.49	21.63	22.97

Table 3: Results for the test dataset (tasks 1 and 2).

Results / Events	**(f2m1)**			**(f2m6)**		
	p	**r**	**fm**	**p**	**r**	**fm**
prot. catab.	78.6	55.0	64.7	71.4	55.6	65.5
phosphoryl.	49.6	56.1	52.7	46.0	55.2	50.2
transcript.	48.9	19.8	28.1	38.7	29.6	33.5
neg. reg.	9.8	7.9	8.8	7.9	7.7	7.8
pos. reg.	10.0	6.6	7.9	10.2	8.0	9.0
regulation	8.6	4.5	5.9	7.5	5.3	6.3
localizat.	28.2	42.9	34.0	23.3	48.9	33.3
gene expr.	51.8	55.1	53.4	52.6	61.2	56.6
binding	19.5	12.1	14.9	22.4	14.4	17.5

Table 4: Results by event for Task 2 on test dataset.

Events and arguments in different sentences of the text (2): as we already discussed in section 2.3, our arguments searching strategy is restricted to the boundaries of the sentence. Some examples of this mistake may be found in document 10395645 in which two events of the token "activation [1354-1364]" is mapped to the themes "caspase-6 [1190-1199]" and "CPP32 [1165-1170]", both located in a different sentence. This mistake usually affects only the false negatives but may cause also a false positive if the system happens to find a valid (wrong) argument in the same sentences for the event under consideration.

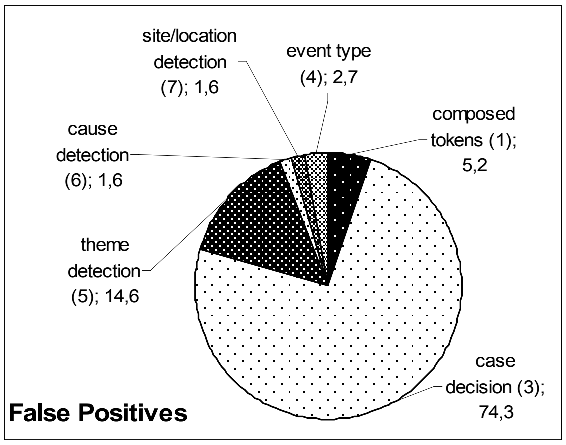


Figure 7: Percent contribution of each error to the false positives.

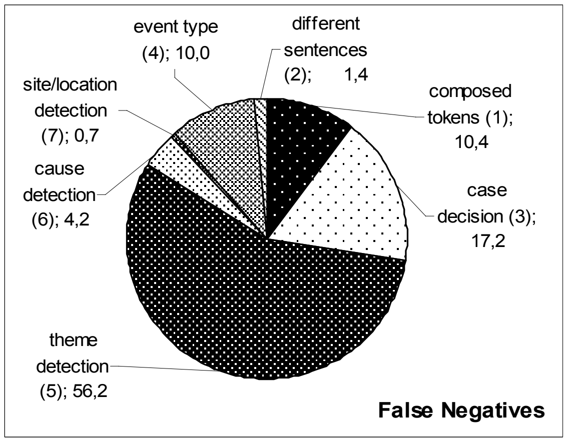


Figure 8: Percent contribution of each error to the false negatives.

Decision for a case (3): this class of error is due to the selection of a wrong case-solution and we include in this class mistakes due to two situations: when the system fails to find any case-solution for an event token (false negative) or when a case-solution is found for a non-event token (false positive). The first situation is only dependent of the searching strategy and its two parameters (MMF and MFC) while the second one is also related to the post-processing step, if the latter succeeds to find a theme for the incorrectly extracted event. An example of a false negative that falls in this group is "dysregulation [727-740]" from document 10229231 that failed to be mapped to a case-solution. Regarding the false positives, this class of mistake is the majority of them and it is due to the low precision of the system that frequently is able to find cases-solution associated to tokens that are not events at all, such as the token "transcript [392-402]" of document 10229231. It should be noted that the incorrect association of a token to a case-solution does not result in a false positive a priori, but only if the post-processing step happen to find a valid theme to it, a mistake further described in group 5.

Wrong type of the event (4): this class of mistake is also due to the wrong selection of a case-solution, but the difference here is that the token is really an event, but the case-solution is of the wrong type, i.e. it has a wrong value for the eventType feature. The causes of this mistake are many, such as, the selection of features (cf. Table 1) or the value of the MFC parameter that may lead to the selection of a wrong but more frequent case. We also include in this group the few false negatives mistakes in which a token is associated to more than one type of event in the gold-standard, such as the token "Overexpression [475-489]" from document 10229231 that is associated both to a Gene Expression and to a Positive Regulation event. One way of overcome it would be to allow the system to associated more than one case to a token, taking the risk of decreasing the precision.

Theme detection (5): in this group falls more than half of the false negatives and we include here only those mistakes in which the token was correctly associated to a case-solution of the correct type. These mistakes may be due to a variety of situations related to the theme detection, such as: the association of the event to another event when it should have been done to a protein or vice-versa (for the regulation events); the mapping of a binding event to one theme only when it should have been two theme or vice-versa; the association of the event to the wrong protein theme, especially when there is more than one nearby; and even not being able to find any theme at all. Also, half of theses mistakes happen when an event is associated to more than one theme separately, not as a second theme. For example, the token "associated [278-288]", from document 10196286, is associated in the gold standard to three themes – "tumor necrosis factor receptor-associated factor (TRAF) 1 [294-351]", "2 [353-354]" and "3 [359-360]" – and we were only able to extract the first of them. This is due to the fact that we restrict the system to search only one "first" and one "second" theme for each event.

Cause detection (6): similar to the previous class, these mistakes happens when associating a cause to an event (regulation events only) when there is no cause related to it or vice-versa. For example, in document 10092805, the system has correctly mapped the token "decreases [1230-1239]" to the theme "4E-BP1 [1240-1246]" but also associated to it an inexistent cause "4E-BP2 [1315-1321]". The evaluation of Task 2 does not allow the partial evaluation of an event and therefore a false positive and a false negative would be returned for the example above.

Site/Location detection (7): this error is similar to the previous one but related only to binding, phosphorylation and localization events, when the system fails to associate a site or a location to an event or vice-versa. For example, in document 10395671, the token "phosphorylation [1091-1106]" was correctly mapped to the theme "Janus kinase 3 [1076-1090]" but was also associated to an inexistent site "DNA [1200-1203]". Once again, the evaluation of Task 2 does not allow the partial evaluation of the event and a false positive and a false negative would be returned.

We have also carried out an evaluation of our own in order to check the performance of our system only on the extraction the entities (event, site and location), not taking into account the association to the arguments. Table 5 resumes the values of precision, recall and f-measure for each type of term. The high recall confirm that most of the entities were successful extracted although the precision is not always satisfactory, proving that the tagging of the entities is not as hard a task as it is the mapping of the arguments. Additional results and more a detailed analysis of the errors may be found at Moara page[4].

4 Conclusions

Results show that our system has performed relatively well using a simple methodology of a machine learning based extraction of the entities and manual rules developed for the post-processing step. The analysis of the mistakes presented here confirms the complexity of the tasks proposed but not the extraction of the event terms (cf. Table 5).

We consider that the part of our system that requires most our attention is the retrieval of the case-solution and the theme detection of the post-processing step, in order to increase the precision and recall, respectively. The decision of searching for a second theme and of associating a single event separately to more than one theme is hard to be accomplished by manual rules and could better be learned automatically using a machine learning algorithm.

Events	**(f2m1)**			**(f2m6)**		
	p	**r**	**fm**	**p**	**r**	**fm**
prot. catab.	70.8	89.5	79.1	69.6	84.2	76.2
phosphoryl.	75.0	94.7	83.7	79.1	89.5	84.0
transcript.	22.7	75.9	34.9	36.4	74.6	48.9
neg. reg.	26.4	56.5	36.0	25.3	43.5	32.0
pos. reg.	24.3	63.7	35.2	26.5	59.1	36.6
regulation	20.8	65.9	31.7	22.1	52.5	31.1
localizat.	47.7	79.5	59.6	49.1	66.7	56.5
gene expr.	46.5	83.4	59.7	50.8	80.2	62.2
binding	29.7	71.1	41.9	29.7	64.4	40.7
entity	12.5	55.3	20.4	16.8	50.0	25.1
TOTAL	**27.5**	**69.2**	**39.4**	**30.9**	**62.9**	**41.4**

Table 5: Evaluation of the extraction of the event and site/location entities for the development dataset.

The automatic analysis of the false positive and false negative mistakes is a hard task since no hint is given for the reason of the mistake by the evaluation system, if due to the event type or to wrong theme, an incorrectly association to an event or even a missing cause or site.

Acknowledgments

This work has been partially funded by the Spanish grants BIO2007-67150-C03-02, S-Gen-0166/2006, PS-010000-2008-1, TIN2005-5619. APM acknowledges the support of the Spanish Ramón y Cajal program. The authors acknowledge support from Integromics, S.L.

References

Aamodt, A., & Plaza, E. (1994). Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. *AI Communications, 7*(1), 39-59.

[4] http://moara.dacya.ucm.es/results_shared_task.html

Chatr-aryamontri, A., Ceol, A., Palazzi, L. M., Nardelli, G., Schneider, M. V., Castagnoli, L., et al. (2007). MINT: the Molecular INTeraction database. *Nucleic Acids Res, 35*(Database issue), D572-574.

Kerrien, S., Alam-Faruque, Y., Aranda, B., Bancarz, I., Bridge, A., Derow, C., et al. (2007). IntAct--open source resource for molecular interaction data. *Nucleic Acids Res, 35*(Database issue), D561-565.

Kim, J.-D., Ohta, T., Pyysalo, S., Kano, Y., & Tsujii, J. i. (2009). *Overview of BioNLP'09 Shared Task on Event Extraction.* Paper presented at the Proceedings of Natural Language Processing in Biomedicine (BioNLP) NAACL 2009 Workshop, Boulder, CO, USA.

Kim, J. D., Ohta, T., & Tsujii, J. (2008). Corpus annotation for mining biomedical events from literature. *BMC Bioinformatics, 9*, 10.

Krallinger, M., Leitner, F., Rodriguez-Penagos, C., & Valencia, A. (2008). Overview of the protein-protein interaction annotation extraction task of BioCreative II. *Genome Biol, 9 Suppl 2*, S4.

Neves, M., Chagoyen, M., Carazo, J. M., & Pascual-Montano, A. (2008). *CBR-Tagger: a case-based reasoning approach to the gene/protein mention problem.* Paper presented at the Proceedings of the BioNLP 2008 Workshop at ACL 2008, Columbus, OH, USA.

Porter, M. (1980). An algorithm for suffix stripping. *Program, 14*(3), 130-137.

Pyysalo, S., Ginter, F., Heimonen, J., Bjorne, J., Boberg, J., Jarvinen, J., et al. (2007). BioInfer: a corpus for information extraction in the biomedical domain. *BMC Bioinformatics, 8*, 50.

Tsuruoka, Y., Tateishi, Y., Kim, J.-D., Ohta, T., McNaught, J., Ananiadou, S., et al. (2005). *Developing a Robust Part-of-Speech Tagger for Biomedical Text.* Paper presented at the Advances in Informatics - 10th Panhellenic Conference on Informatics.

Biomedical Event Annotation with CRFs and Precision Grammars

Andrew MacKinlay, David Martinez and **Timothy Baldwin**
NICTA Victoria Research Laboratories
University of Melbourne, VIC 3010, Australia
{amack,davidm,tim}@csse.unimelb.edu.au

Abstract

This work describes a system for the tasks of identifying events in biomedical text and marking those that are speculative or negated. The architecture of the system relies on both Machine Learning (ML) approaches and hand-coded precision grammars. We submitted the output of our approach to the event extraction shared task at BioNLP 2009, where our methods suffered from low recall, although we were one of the few teams to provide answers for task 3.

1 Introduction

We present in this paper our techniques for the tasks 1 and 3 of the event extraction shared task at BioNLP 2009. We make use of both Machine Learning (ML) approaches and hand-coded precision grammars in an architecture that combines multiple dedicated modules. In the third task on negation/speculation, we extract extract rich linguistic features resulting from our HPSG high-precision grammar to train an ML classifier.

2 Methodology

2.1 Task 1: Shallow Features and CRFs

Our system consists of two main modules, the first of which is devoted to the detection of event trigger words, and the second to event–theme analysis.

2.1.1 Trigger-word detection

We developed two separate systems to perform trigger word detection, and also a hybrid system which combines their outputs. The first system is a simple dictionary-based look-up tagger; the second system learns a structured model from the training data using conditional random fields (CRFs). For pre-processing, we relied on the domain-specific token and sentence splitter from the JULIE Lab (Tomanek et al., 2007) and the GENIA tagger for lemmatisation, POS tagging, chunking, and protein detection (Tsuruoka et al., 2005).

The look-up tagger operates by counting the occurrences in the training data of different event tags for a given term. Over the development and test data, each occurrence of a given term is assigned the event class with the highest prior in the training data. We experimented with a frequency cut-off that allows us to explore the precision/recall trade-off.

Our second system relies on CRFs, as implemented in the CRF++ toolkit (Lafferty et al., 2001). CRFs provide a discriminative framework for building structured models to segment and label sequence data. CRFs have the well-known advantage that they both model sequential effects and support the use of large numbers of features. In our experiments we used the following feature types: word-forms, lemmas, POS, chunk tags, protein annotation, and grammatical dependencies. For dependency annotation, we used the Bikel parser and GDep as provided by the organisers. This information was provided as a feature that expresses the grammatical function of the token. We explored window sizes of ± 3 and ± 4.

Finally, we tested combining the outputs of the look-up tagger and CRF, by selecting all trigger words from both outputs.

Proceedings of the Workshop on BioNLP: Shared Task, pages 77–85,
Boulder, Colorado, June 2009. ©2009 Association for Computational Linguistics

2.1.2 Event-theme construction

We constructed the output for task 1 by differentiating among three types of events, according to their expected themes: basic events, binding events, and regulation events. We applied a simple strategy, assigning the closest events or proteins within a given sentence as themes.

For the basic events, we simply assigned the closest protein, an approach that we found to perform well over the training and development data. For binding events, we estimated the maximum distance away from the event word(s) for themes, and the maximum number of themes. For regulation events, we had to choose between proteins or events as themes, and the CAUSE field was also required. Again, we relied on a maximum distance threshold, and gave priority to events over proteins as themes. We removed regulation events as theme candidates, since our basic approach could not indicate the direction of the regulation. We also tested predicting the CAUSE by relying on the protein closest to the regulation event.

2.2 Task 3: Deep Parsing and Maximum Entropy classification

For task 3 we ran a syntactic parser over the abstracts and used the outputs to construct feature vectors for a machine learning algorithm. We built two classifiers (possibly with overlapping sets of feature vectors) for each training run: one to identify speculation and one for negation. We deliberately built a separate binary classifier for each task instead of a single four-class classifier, since the problem naturally decomposes this way. Speculation and negation are independent of one another (informally, not statistically) and it enables us to focus on feature engineering for each subtask.

2.2.1 Deep Parsing with the ERG

It seemed likely that syntactico-semantic analysis would be useful for task 3. To identify negation or speculation with relatively high precision, it is probable that knowledge of the relationships of possibly distant elements (such as the negation particle *not*) to a particular target word would provide valuable information for classification.

Further to this, it was our intention to evaluate the utility of deep parsing in such an approach, rather than a shallower annotation such as the output of a dependency parser. With this in mind, we selected the English Resource Grammar[1] (ERG: Copestake and Flickinger (2000)), an open-source, broad-coverage high-precision grammar of English in the HPSG framework.

While the ERG is relatively robust across different domains, it is a general-purpose resource, and there are some aspects of the language used in the biomedical abstracts that cause difficulties; unknown word handling is especially important given the nature of terms in the domain. Fortunately we can make some optimisations to mitigate this. The GENIA tagger mentioned in Section 2.1.1 provides both POS and named entity annotations, which we used to constrain the input to the ERG in two ways:

- Biological named entities identified by the GENIA tagger are flagged as such, and the parser does not attempt to decompose them.

- POS tags are appended to each input token to constrain the token to an appropriate category if it is absent from the ERG lexicon.

With these modifications to the parser, as well as preprocessing to handle differences in the tokenisation expected by the ERG to the output of the tagger, we were able to obtain a spanning parse for 72% of the training sentences. This still leaves 28% of the sentences inaccessible – the need for a fallback strategy is discussed further in Section 4.2.

2.2.2 Feature Extraction from RMRSs

Rather than outputting syntactic parse trees, the ERG can also produce output in particular semantic formalisms: Minimal Recursion Semantics (MRS: Copestake et al. (2005)) and the closely related Robust Minimal Recursion Semantics (RMRS: Copestake (2004)). For our feature generation here we make use of the latter.

Figure 1 shows an example RMRS obtained from one of the training documents. While there is insufficient space to give a complete treatment here, we highlight several aspects for expository purposes.

[1]Specifically the July 2008 version, downloadable from `http://lingo.stanford.edu/ftp/test/`

```
l1,
{ l3: _thus_a_1⟨62:67⟩(e5, ARG1: h4),
  l16: generic_unk_nom_rel⟨68:78⟩(x11, CARG: 'nf- kappa _b'),
  l6: udef_q_rel⟨68:89⟩(x9, RSTR: h8, BODY: h7),
  l10: compound_rel⟨68:89⟩(e12, ARG1: x9, ARG2: x11),
  l13: udef_q_rel⟨68:89⟩(x11, RSTR: h15, BODY: h14),
  l101: _activation_n_1⟨79:89⟩(x9),
  l17: neg_rel⟨94:97⟩(e19, ARG1: h18),
  l20: _require_v_1⟨98:106⟩(e2, ARG1: u21, ARG2: x9),
  l102: parg_d_rel⟨98:106⟩(e22, ARG1: e2, ARG2: x9),
  l103: _for_p⟨107:110⟩(e24, ARG1: e2, ARG2: x23),
  l34: generic_unk_nom_rel⟨111:129⟩(x29,
    CARG: 'neuroblastoma cell'),
  l25: udef_q_rel⟨111:146⟩(x23, RSTR: h27, BODY: h26),
  l28: compound_rel⟨111:146⟩(e30, ARG1: x23, ARG2: x29),
  l31: udef_q_rel⟨111:146⟩(x29, RSTR: h33, BODY: h32),
  l104: _differentiation_n_of⟨130:146⟩(x23, ARG1: u35) },
{ h4 qeq l17, h8 qeq l10, h15 qeq l16, h18 qeq l20, h27 qeq l28,
    h33 qeq l34 },
{ l10 in-g l101, l20 in-g l102, l20 in-g l103, l28 in-g l104 }
```

Figure 1: RMRS representation of the sentence *Thus NF-kappa B activation requires neuroblastoma cell differentiation* showing, in order, elementary predicates, qeq-constraints, and in-g constraints

The primary component of an RMRS is bag of *elementary predicates*, or EPs. Each EP shown has: (a) a label, such as 'l104'; (b) a predicate name, such as '_differentiation_n_1' (where 'n' indicates the part-of-speech); (c) character indices to the source sentence; and (d) a set of arguments. The first argument is always *ARG0* and is afforded special status, generally referring to the variable introduced by the predicate. Subsequent arguments are labelled according to the relation of the argument to the predicate. Arguments can be variables such as 'e30' or 'x23' (where the first letter indicates the nature of the variable – 'e' referring to events and 'x' to entities), or *handles* such as 'h33'.

These handles are generally used in the *qeq constraints*, which relate a handle to a label, indicating a particular kind of outscoping relationship between the handle and the label – either that the handle and label are equal or that the handle is equal to the label except that one or more quantifiers occur between the two (the name is derived from 'equality module quantifiers'). Finally there are in-g constraints which indicate that labels can be treated as equal. For our purposes this simply affects which qeq constraints they participate in – for example from the in-g constraint 'l28 in-g l104' and the qeq constraint 'h27 qeq l28', we can also infer that 'h27 qeq l104'.

In constructing features, we make use of:

- The *outscopes* relationship (specifically qeq-outscopes) – if EP A has a handle argument which qeq-outscopes the label of EP B, A is said to immediately outscope B; *outscopes* is the transitive closure of this.

- The *shared-argument* relationship, where EPs C and D refer to the same variable in one or more of their argument positions. We also in some cases make further restrictions on the types of arguments (*ARG0*, *RSTR*, etc) that may be shared on either end of the relationship.

2.2.3 Feature Sets and Classification

Feature vectors for a given event are constructed on the basis of the trigger word for the particular event, which we assume has already been identified; a natural consequence is that all events with the same trigger words have identical feature vectors. We use the term *trigger EPs* to describe the EP(s) which correspond to that trigger word – i.e. those whose character span encompasses the trigger word. We have a potentially large set of related EPs (with the kinds of relationships described above), which we filter to create the various feature sets, as outlined below.

We have several feature sets targeted at identifying negation:

- NEGOUTSCOPE2: If any EPs in the RMRS have predicate names in {*_no_q*, *_but+not_c*, *_nor_c*, *_only_a*, *_never_a*, *_not+as+yet_a*, *_not+as+yet_a*, *_unable_a*, *neg_rel*}, and that EP outscopes a trigger EP, set a general feature as well as a specific one for the particle.

- NEGCONJINDEX: If any EPs in the RMRS have predicate names in {*_not_c*, *_but+not_c*, *_nor_c*}, the *R-INDEX* (RHS of a conjunction) of that EP is the *ARG0* a trigger EP, set a general feature as well as a specific one for the particle – capturing the notion that these conjunctions are semantically negative for the particle on the right. This also had a corresponding feature for the *L-INDEX* of *_nor_c*, corresponding to the LHS of the *neither...nor* construction.

- ARG0NEGOUTSCOPEESA: For any EPs which have an argument that matches the *ARG0* of a trigger EP, if they are outscoped by an EP whose predicate name is in the list { *_only_a*, *_never_a*, *_not+as+yet_a*, *_not+as+yet_a*, *_unable_a*, *neg_rel*}, set a general feature to true, as well as features for the name of the outscoping and outscoped EPs. This is designed to catch trigger EP which are nouns, where the verb of which they are subject or object (or indeed an adjective/preposition to which they are linked) is semantically negated.

And several targeted at identifying speculation:

- SPECVOBJ2: if a verb is a member of the set {*_investigate*, *_study*, *_examine*, *_test*, *_evaluate*, *_observe*} and its *ARG2* (which corresponds to the verb object) is the *ARG0* of a trigger EP. This has a general feature for if any of the verbs match, and a feature which is specific to each verb in the target list.

- SPECVOBJ2+WN: as above, but augment the list of seed verbs with a list of WordNet sisters (i.e. any lemmas from any synsets for the verb), and add a feature which is set for the seed verbs which gave rise to other sister verbs.

- MODALOUTSCOPE: modal verbs (*can*, *should*, etc) may be strong indicators of speculation; this sets a value when the trigger EP is outscoped by any predicate corresponding to a modal, both as a general feature and a specific feature for the particular modal.

- ANALYSISSA: the *ARG0* of the trigger EP is also an argument of an EP with the predicate name *_analysis_n*. Such constructions involving the word *analysis* are relatively frequent in speculative events in the data.

And some general features, aiming to see if the learning algorithm could pick up other patterns we had missed:

- TRIGPREDPROPS: Set a feature value for the predicate name of each trigger EP, as well as the POS of each trigger EP.

- TRIGOUTSCOPES: Set a feature value for the predicate name and POS of each EP that is outscoped by the trigger EP.

- MODADJ: Set a feature value for any EPs which have an *ARG1* which matches the *ARG0* of the trigger EP if their POS is marked as adjective or adverb.

- +CONJ: This is actually a variant on the feature extraction method, which attempts to abstract away the effect of conjunctions. If the trigger EP is a member of a conjunction (i.e. shares an *ARG0* with the *L-INDEX* or *R-INDEX* of a conjunction), also treat the EPs which are conjunction parents (and their conjunctive parents if they exist) as trigger EPs in the feature construction.

2.2.4 Implementation

To produce training data to feed into a classifier, we parsed as many sentences as possible using the ERG, and used the output RMRSs to create training data using various combinations of the feature sets described above. The construction of features, however, presupposes annotations for the events and trigger words. For producing training data, we used the provided trigger annotations. For the test phase, we simply use the outputs of the classifier we built in phase 1, selecting the combination with the best performance over the development set. This pipeline architecture places limits on annotation performance – in particular, the recall in task 1 is an upper bound on task 3 recall. We used a maximum entropy classification algorithm for the ML component here – it has a low level of parameterization and is a solid performer in NLP tasks. The implementation we used was Zhang Le's Maxent Toolkit.[2]

3 Development experiments

3.1 Task 1

We devised a set of experiments over the trial, training, and development data in order to estimate the parameters for our final submission. Using the trial data, we performed manual error analysis on the rules used to construct events. With the training

[2]http://homepages.inf.ed.ac.uk/s0450736/maxent_toolkit.html

data, we performed our own evaluation based on cross-validation to detect trigger words and construct events. For the experiments over the development data, we relied on the evaluation interface provided by the organisation. We focused on testing the following modules: look-up tagger, CRF, combined system, and event construction.

First, we tuned the parameters of our look-up tagger over the training data. We used a threshold on the minimum number of term occurrences required to use the class information for that term from the training data. We evaluated thresholding on raw frequencies, and also on the percentage of occurrences of the term that were linked to the majority event. In cross-validation over the training data, we found that the raw-frequency threshold worked best, achieving a maximum F-score of 38.86%, as compared to 30.81% for the percentage approach (the results are shown in the bottom part of Table 1). We also estimated the frequency threshold as ≥ 25, and observed that most of the terms identified consisted of a single word, due to data sparseness in the training set.

Our next experiments are devoted to the CRF system, focusing on feature engineering. The results over the training data for: (a) the full feature set, and (b) removing one feature type at a time, are shown in Table 1, for windows of size ±3 and ±4. We can see that the best F-score is achieved by the ±3 word-window system when removing the syntactic dependencies from Bikel's parser. These results improved over the look-up system.

As a final experiment on feature combinations and window size, we used the development evaluation interface. We submitted the best combinations shown in the above experiment, and also syntactic dependencies extracted with GDep. We observed the same behaviour as in training data, with the ±3 word window obtaining the best F-score, and syntactic dependencies harming performance. These results are shown in the upper part of Table 2. Our final CRF system used this configuration (±3 word window and all feature types except syntactic dependencies).

Our next step was to test the integration of the look-up tagger and CRF into a single system. We observed that by combining the outputs directly we could improve over the recall of CRF, and achieve higher F-score. This approach is referred to as "Combined (All)" in Table 2. We also tested the results when choosing either the look-up tagger or CRF depending on their performance over each event in the training data. The results of this system ("Combined (Best)") show a slight improvement over the basic combination.

Finally, we analysed the results of the event construction step. We used the gold-standard trigger annotation over the trial data and analysed the errors of our rules. We found out that there were three main types of error: (1) incorrect assignation of regulation themes; (2) trigger words having multiple themes; and (3) themes crossing sentence boundaries. We plan to address these problems in future work. We also observed that predicting CAUSE for the regulatory events caused the F-score to drop, resulting in us removing this functionality from the system.

W. size	Feats.	Rec.	Prec.	FSc.
±3	All	30.28	64.44	41.20
±3	−synt. dep.	30.20	65.01	**41.24**
±3	−protein NER	28.04	65.73	39.31
±3	−chunking	30.13	65.16	41.20
±3	−POS	29.68	65.25	40.80
±3	−lemma	27.96	62.60	38.66
±3	−word form	29.98	63.81	40.79
±4	All	28.86	66.15	40.19
±4	−synt. dep.	29.75	**67.06**	41.22
±4	−protein NER	28.11	66.73	39.56
±4	−chunking	28.56	66.61	39.98
±4	−POS	28.19	66.67	39.62
±4	−lemma	26.55	65.20	37.73
±4	−word form	28.19	65.28	39.38
Look-up (freq.)		**52.14**	30.97	38.86
Look-up (perc.)		38.20	25.82	30.81

Table 1: Trigger-word detection performance over training data. Results for the look-up tagger and CRFs with the full feature set and when removing one feature type at a time, for 3 and 4 word windows. The best results per column are shown in bold.

W. size	Feats.	Rec.	Prec.	FSc.
±3	All - synt.	17.55	56.17	26.75
±4	All - synt.	17.38	56.75	26.62
±3	All (GDep)	15.48	**58.69**	24.50
Combined (All)		**26.94**	27.83	27.38
Combined (Best)		21.24	39.92	**27.73**

Table 2: Performance of selected feature and window-size combinations over development data. Best results per column are given in bold.

N1:	NegOutscope2+Conj, NegConjIndex
N2:	*N1*, TrigPredProps
N3:	*N1*, Arg0NegOutscopeeSA
N4:	*N3*, TrigPredProps, NegVOutscope
N5:	*N3*, NegVOutscope
S1:	SpecVObj2+WN+Conj, AnalysisSA
S2:	*S1*, TrigPredProps
S3:	*S1*, ModAdj, ModalOutscope
S4:	*S3*, TrigOutscopes
S5:	SpecVObj2+WN+Conj, ModAdj, ModalOutscope,TrigOutscopes
B^{+y}_{-x}:	Context window of lemmatized tokens: x preceding and y following.

Table 3: Task 3 feature sets

Task 1	Mod	Feats.	Rec.	Prec.	FSc.
Gold	Spec	B^{+2}_{-2}	23.2	40.0	29.3
Gold	Spec	B^{+3}_{-3}	22.1	47.7	30.2
Gold	Spec	S2	15.8	83.3	26.5
Gold	Spec	S3	18.9	78.3	30.5
Gold	Spec	S3,B^{+2}_{-2}	21.1	58.8	31.0
Gold	Spec	S3,B^{+3}_{-3}	23.2	57.9	33.1
Comb(best)	Spec	S3	4.2	21.0	7.0
Gold	Spec	S4	17.9	94.4	30.1
Gold	Spec	S5	17.9	100.0	30.4
Gold	Neg	B^{+0}_{-2}	14.0	33.3	19.7
Gold	Neg	B^{+1}_{-3}	15.0	30.2	20.0
Gold	Neg	N2	19.6	61.8	29.8
Comb(best)	Neg	N2	0.9	7.7	1.7
Gold	Neg	N3	15.9	68.0	25.8
Gold	Neg	N4	19.6	67.7	30.4
Gold	Neg	N4,B^{+1}_{-3}	22.4	52.2	31.4
Gold	Neg	N4,B^{+0}_{-2}	24.3	68.4	35.9
Gold	Neg	N5	16.8	69.2	30.1

Table 4: Results (exact match) over development data for task 3 using gold-standard event/trigger annotations and selected other annotations for task 1. Feature sets described in Table 3

3.2 Task 3

We evaluated the classification performance of various feature sets (including some not described here) using 10-fold cross-validation over the training data in the initial stages. We ran various combinations of the most promising features over the development data and evaluated their relative performance in an attempt to avoid overfitting.

To evaluate the performance boost we got in task 3 relative to more naive methods, we also experimented with feature sets based on a bag-of-words approach with a sliding context window of lemmatised tokens on either side. We evaluated all combinations of preceding and following context window sizes from 0 to 3. There are features for tokens that precede the trigger, follow the trigger, or lie anywhere within the context window, as well as for the trigger itself. A 'token' here may also be a named biological entity (protein etc) produced by GENIA tagger in our preprocessing phase, which would not be lemmatised. For comparability we only evaluate these features for sentences which we were able to parse. For the best performing baseline and RMRS-based feature sets, we also tested them in combination to see whether the features produced were complementary.

In Table 4 we present the results over the development data, using the provided gold-standard annotations of trigger words, as well as some selected results for our other task 1 outputs. The gold-standard figures are unrealistically high compared to what we would expect to achieve against the test data, but they are indicative at least of what we could achieve with a perfect event classifier. Similar to task 1, our system shows reasonable precision but suffers badly in recall. The substantially poorer performance when using our own annotations for the input events is discussed in more detail in Section 4.2

One area where we could improve is to go after the 30% of sentences for which we do not have a spanning parse and resultant RMRS. To reuse existing infrastructure, we could produce RMRS output from an alternative processing component with broader coverage but less precision. Several methods exist to do this – e.g. producing RMRS output from RASP (Briscoe et al., 2006) is described in Frank (2004). However there is clearly room for improvement in the remaining 70% of sentences which we can parse – our results in Table 4 are still well below the limit of roughly 70% recall.[3]

Additional lexical resources beyond WordNet, particularly domain-specific ones, are likely to be useful in boosting performance since they will help maximally utilise the training data. Additionally, we have not yet made use of other event annotations apart from the trigger words – features based on characteristics such as the event class or properties of the event arguments could also be useful.

[3]We have not performed any analysis to verify whether the number of events per sentence differs between parseable and unparseable sentences.

System	Rec.	Prec.	FSc.
Combined (Best)	**17.44**	**39.99**	**24.29**
Combined (All)	24.36	30.87	27.23
CRF	12.23	62.24	20.44
CRF (+ synt feats)	12.01	61.91	20.11
Look-Up	22.88	29.67	25.84
Look-Up (freq >= 20)	23.26	26.74	24.88
Look-Up (freq >= 30)	21.37	30.50	25.13

Table 5: Task 1 results with approximate span matching, recursive evaluation (our final submission is in bold)

Event Class	Rec.	Prec.	FSc.
Localization	25.86	65.22	37.04
Binding	17.00	28.92	21.42
Gene-expression	45.71	69.18	55.05
Transcription	34.31	26.26	29.75
Protein-catabolism	42.86	85.71	57.14
Phosphorylation	45.19	64.21	53.04
EVT-TOTAL	35.84	53.15	42.81
Regulation	15.46	13.24	14.26
Positive-regulation	13.84	14.82	14.31
Negative-regulation	12.14	20.44	15.23
REG-TOTAL	13.73	15.31	14.48
ALL-TOTAL	24.36	30.87	27.23

Table 6: Results for the different events from our combined system. Averaged scores for single events, regulations, and all.

4 Results

4.1 Task 1

Our experiments on the training and development set showed that our CRF++ was biased towards precision at the cost of recall, and for the look-up system the best F-score was obtained when aiming for high recall at the cost of lower precision. The best results were obtained when combining both approaches, and this was the composition of the system we submitted.

For our final submission, the CRF++ approach had a ± 3 word window, and all the features except for syntactic dependencies, which were found to harm performance. Our final look-up system relied on raw frequencies to choose candidate terms, and those above 24 occurrences in training data were included in the dictionary. For the combination, we observed that for most events the look-up system performed better (although the overall F-score was lower), and we decided to use the CRF++ output only for the events that showed better performance than the look-up system (TRANSCRIPTION, GENE EXPRESSION, and POSITIVE REGULATION).

The results over the test data for our final submission and the main variants we explored are shown in Table 5. We can see that the CRF performed poorly, with very low recall over the test set, in contrast with the development results, where the higher recall resulted in a higher F-score than the look-up approach. The best of our systems was the full combination of CRF and the look-up tagger, with a 27.23% F-score.

The results for each event separately are given in Table 6. The system performs much worse on regulation events, due to the difficulty of having to correctly identify other events in the near context.

4.2 Task 3

For testing, we repurposed all of the development data as training data and retrained our classifiers. The results in Table 7 were somewhat disappointing, but a drop in recall versus the equivalent run over the development data using oracle task 1 annotations was unsurprising and the ratio of this drop is within the bounds of what we would expect. The substantial drop in precision can similarly be explained by flow-on effects from our task 1 classification, a natural consequence of our pipeline architecture. It is quite possible for our system to identify false positive events as being modified; in the online evaluation system, these classifications of non-existent events reduce our precision in task 3.

In the feature engineering stage, we primarily used the oracle data for task 1 to maximise the amount of training data available. We felt that if we were to use our task 1 classifications for events and trigger words, the effectively lower number of training instances would only hurt performance. However this possibly led to bias towards features which were more useful for classifying events that we couldn't successfully classify in task 1. The development set shows similar performance drops under these conditions in Table 4.

It is also possible that our features work reasonably but that our classification engine trained over the oracle data simply learnt the wrong parameters

Task 1	**Mod**	**Fts.**	**Rec.**	**Prec.**	**FSc.**
Comb(Best)	Spc	B_{-3}^{+3}	2.88	12.24	4.67
Comb(Best)	Spc	S2	4.33	37.50	7.76
Comb(Best)	Spc	**S3**	**4.81**	**30.30**	**8.30**
Comb(All)	Spc	S3	5.29	26.19	8.80
Comb(Best)	Spc	S3,B_{-3}^{+3}	4.81	14.08	7.17
Comb(Best)	Spc	S4	3.85	27.59	6.75
Comb(Best)	Spe	S5	3.85	27.59	6.75
Comb(Best)	Neg	B_{-1}^{+3}	3.96	25.00	6.84
Comb(Best)	Neg	**N2**	**5.29**	**34.48**	**9.17**
Comb(All)	Neg	N2	5.73	30.00	9.62
Comb(Best)	Neg	N3	5.29	27.78	8.88
Comb(Best)	Neg	N4	5.29	34.48	9.17
Comb(Best)	Neg	N4,B_{-2}^{+0}	4.85	28.12	8.27
Comb(All)	Neg	N4	5.73	27.27	9.47
Comb(Best)	Neg	N5	5.29	29.41	8.96

Table 7: Results over test data for task 3 using gold-standard event annotations (approx recursive matching), showing which set of trigger word classifications from task 1 was used as input (submitted results in bold). Feature sets described in table 3

for the events we had identified correctly in task 1. We could check this by training a classifier using our task 1 event classifications combined with the gold-standard trigger annotations. However combining the gold-standard annotations for task 3 with the classifier outputs of task 1 is non-trivial and was not attempted due to time constraints. It also would have been instructive to calculate a ceiling on our task 3 performance given our performance in task 1 – i.e. how many modifications we could have correctly identified with a perfect task 3 classifier, but we were not able to show this for similar reasons.

5 Conclusions

Our analysis of task 1 seemed to indicate that the scarcity of training instances was the main reason for the low recall of CRFs. The look-up system contributed to increase the recall, but at the cost of lower precision. In order to improve this module we plan to find ways to extend the training data automatically in a bootstrapping process.

Another limitation of our system is the event-construction module, which follows simple rules and performs poorly on regulation events. For this subtask we plan to extend the rule set and apply optimisation techniques, following the lessons learned in error analysis.

In task 3 we investigated the application of a precise, general-purpose grammar over this domain, and were relatively successful. However, while the parse coverage for task 3 is very respectable for a precision grammar on comparatively difficult material, it is clearly unwise to throw away 30% of sentences, so a method to extract features from these is desirable. Further sources of data would also be useful, such as data from the event annotations themselves, and additional lexical resources tailored to the biomedical domain.

We have also shown the syntactico-semantic output of a deep parser, in the form of an RMRS, can be beneficial in such a task compared with a more naive approach based on bags of words within a sliding context window. From Table 4, for negation, the syntactic features provided substantial performance gains over the best set of baseline parameters we could find. For speculation the evidence here is less compelling, with similar scores from both approaches. Over test data in Table 7, the the deep methods showed superior performance, albeit over a smaller number of instances. Regardless, the RMRS still has some advantages, giving (unsurprisingly) higher precision than the baseline methods. Combining naive and deep features does tend to give slightly higher performance than either of the inputs over the development data (although not over the test data, perhaps due to the poorer performance of naive methods), suggesting that the two approaches identify slightly different kinds of modification.

Our system suffered from the pipeline approach – there was no way to recover from an incorrect classification in task 1, resulting in greatly reduced precision and recall in task 3. It is possible that a carefully constructed integrated system could annotate events for trigger words and argument at the same time as modification, with features shared between the two, which may avoid some of these issues.

Acknowledgements

We wish to thank Rebecca Dridan, Dan Flickinger and Lawrence Cavedon for their advice. NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program.

References

Edward Briscoe, John Carroll, and Rebecca Watson. 2006. The second release of the RASP system. In *Proceedings of the COLING/ACL 2006 Interactive Poster System*, pages 77–80, Sydney, Australia.

Ann Copestake and Dan Flickinger. 2000. An open source grammar development environment and broad-coverage English grammar using HPSG. In *International Conference on Language Resources and Evaluation*.

Ann Copestake, Dan Flickinger, Ivan A. Sag, and Carl Pollard. 2005. Minimal recursion semantics: An introduction. *Research on Language and Computation*, pages 281–332.

Ann Copestake. 2004. Report on the design of RMRS. Technical Report D1.1a, University of Cambridge, Cambridge, UK.

Anette Frank. 2004. Constraint-based RMRS construction from shallow grammars. In *COLING '04: Proceedings of the 20th international conference on Computational Linguistics*, page 1269, Morristown, NJ, USA. Association for Computational Linguistics.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the International Conference on Machine Learning*, pages 282–289.

Katrin Tomanek, Joachim Wermter, and Udo Hahn. 2007. Sentence and token splitting based on conditional random fields. In *Proceedings of the 10th Conference of the Pacific Association for Computational Linguistics*, pages 49–57, Melbourne, Australia.

Yoshimasa Tsuruoka, Yuka Tateishi, Jin-Dong Kim, Tomoko Ohta, John McNaught, Sophia Ananiadou, and Jun'ichi Tsujii. 2005. Developing a robust part-of-speech tagger for biomedical text. In *Advances in Informatics - 10th Panhellenic Conference on Informatics*, pages 382–392.

Molecular event extraction from Link Grammar parse trees

Jörg Hakenberg[1], Illés Solt[2], Domonkos Tikk[2,3], Luis Tari[1],
Astrid Rheinländer[3], Quang Long Ngyuen[3], Graciela Gonzalez[1], and Ulf Leser[3]
[1]Arizona State University, Tempe, AZ 85283, USA,
[2]Budapest University of Technology and Economics, 1117 Budapest, Hungary,
[3]Humboldt Universität zu Berlin, 10099 Berlin, Germany.

Abstract

We present an approach for extracting molecular events from literature based on a deep parser, using in a query language for parse trees. Detected events range from gene expression to protein localization, and cover a multitude of different entity types, including genes/proteins, binding sites, and locations. Furthermore, our approach is capable of recognizing negation and the speculative character of extracted statements. We first parse documents using Link Grammar (BioLG) and store the parse trees in a database. Events are extracted using a newly developed query language with traverses the BioLG linkages between trigger terms, arguments, and events. The concrete queries are learnt from an annotated corpus. On BioNLP Shared Task data, we achieve an overall F1-measure of 29.6%.

1 Introduction

Biomedical text mining aims at making the wealth of information available in publications available for systematic, automatic studies. An important area of biomedical text mining is concerned with the extraction of relationships between biological entities, especially the extraction of protein–protein interactions from PubMed abstracts (Krallinger et al., 2008). The BioNLP'09 Shared Task addresses the problem of extracting nine different types of molecular events (Kim et al., 2009) and thus targets a problem that is considerable less-well studied than protein-protein interactions. Such molecular events include statements about the expression level of genes, the binding sites of proteins, and the up/down regulation of genes, among others. All events focus on genes/proteins and may include only a single protein (e.g., protein catabolism), multiple proteins (e.g., binding), and other arguments (e.g., phosphorylation *site*; protein *location*). The most complex type of event considered in the task are regulations, which may refer to other events (negative regulation of gene expression) and may also include causes as arguments. The task also addresses the problem that experimental findings often are described in a defensive manner ("Our results suggest ...") or may appear in negated context. This meta-information about an extracted event should be taken into account when text mining results are used in automated analysis pipelines, but recognizing the degree of confidence that can be put into an event adds further complexity to the task. Overall, the three tasks in BioNLP'09 are: *1)* event detection and characterization, *2)* event argument recognition, and *3)* recognition of negations and speculations.

The approach we present in this paper addresses all three tasks. Essentially, our system consists of three components: A deep parser, a query language for parse trees, and a set of queries that extract specific events from parse trees. First, we use the BioLG parser (Pyysalo et al., 2006) for parsing sentences into a graph-like structure. Essentially, BioLG recognizes the syntactic structure of a sentence and represents this information in a tree. It adds links between semantically connected elements, such as the links between a verb and its object and subject. Second, we store the result of BioLG in a relational database. This information is accessed by a special-purpose query language (Tu et al., 2008) that

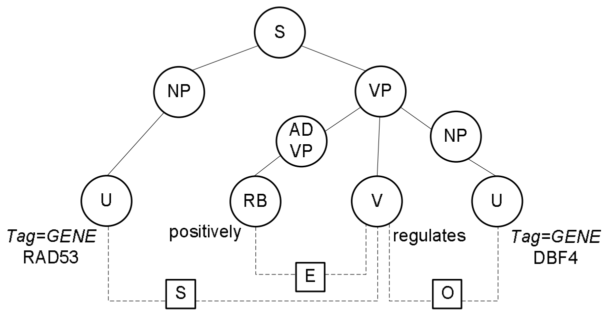


Figure 1: Parse tree where constituents are connected by solid lines, linkages between terminals shown as dotted lines. E: adverb to verb, S: subject to verb, O: verb to object.

matches a user-defined linguistic pattern describing relationships between terms (the query) to the database of stored graphs. The query language thus is a powerful, scalable, extensible, and systematic way of describing extraction patterns. Using these tools, we can solve the BioNLP tasks by means of a set of queries, extracted from the training data set.

2 Methods

The Link Grammar parser is a deep syntactic parser based on the Link Grammar theory (Sleator and Temperley, 1993), which consists of a set of words and linking requirements between words. The particular implementation of Link Grammar parsing we use in our system is the BioLG parser described in Pyysalo et al. (2006), which modifies the original parser by extending its dictionary and by adding more rules for guessing structures when facing unknown words. The output of the parser is twofold: it produces a *constituent tree* as well as a *linkage* that shows the dependencies between words. In Figure 1, solid lines indicate parent-child relationships in the constituent tree, and dotted lines represent the linkage. Three links were detected in the sentence: `S` connects the subject-noun `RAD53` to the transitive verb `regulates`, `O` connects the transitive verb `regulates` to the direct object `DBF4`, and `E` connects the verb-modifying adverb `positively` to the verb `regulates`.

Our detection of arguments for events is based on Link Grammar linkages obtained from training data. Essentially, we automatically extract all shortest link paths that connect event trigger terms to themes, themes to sites, themes to locations, and so on. We

```
           +---------------Jp------------+
           |   +----------CH-------------+
           |   |   +---------CH----------+
+----Mp----+   |   |          +----CH----+
|          |   |   |          |          |
expression of P53, Rb, and Bcl-xL proteins
```

Figure 2: Linkage in a gene expression evidence. Mp: prepositional phrase modifying a noun; Jp: connects preposition to object; CH: noun modifier.

describe these examples as queries against a parse tree, and evaluate these queries on the test data to extract and assemble events. An example for a linkage in a gene expression evidence is shown in Figure 2. It illustrates that the event trigger term ‘expression’ is connected to the three proteins ‘P53’, ‘Rb’, and ‘Bcl-XL’ in exactly the same way.

Our method for event argument recognition is based on three components. The first parses training as well as test data using the BioLG parser, and stores the result in a relational database. The second component is a query language to search the databases for known linkages. The third component extracts these linkages from training data and rewrites into such queries. These components are detailed in Sections 2.1 to 2.3. Section 2.4 explains our methods for context identification with respect to negations and speculations. Sections 2.5 and 2.6, finally, explain how we handle anaphora and enumerations, respectively.

2.1 Parse Tree Database and Query Language

A fundamental component of our approach is a parse tree database (PTDB) for storing and querying parse trees (Tu et al., 2008). PTDB is a relational database for storing the results of the BioLG parser on arbitrary texts. For the task, we parsed all texts from the training, development and testing data set. Recognition of entity types (gene etc.) of word tokens relied on the provided annotation. Each abstract is represented in a manner that captures both the document structure (such as title, sections, sentences) and the parse trees of sentences.

Parse trees in PTDB are accessed by means of a special purpose query language, called PTQL. PTQL is an extension to LPath (Bird et al., 2006), which itself is an adaptation of XPath (XPath, 2009) to linguistic structures. Essentially, a PTQL query is a hierarchical pattern that is matched against a set of constituent trees together with additional requirements on linkages between matches. More specifically, a PTQL query consists four components delimited by colons: *1)* tree pattern, *2)* link conditions, *3)* proximity conditions, and *4)* return expression. A *tree pattern* describes the hierarchical structure and the horizontal order between the nodes of a parse tree, a *link condition* describes the linking dependencies between nodes, a *proximity condition* specifies words that are within a specified number of words in the sentence, and the *return expression* defines which variables should be returned as query result. An example PTQL query is shown in Figure 3.

PTQL queries are evaluated on a PTDB using a two step process. A query is first translated into an IR-style keyword query to efficiently filter out irrelevant sentences. This step is performed outside the database using an inverted index built with Lucene (2009). In the second step, the query is translated into a complex SQL command which is restricted to the sentence IDs that passed the first step. This query is evaluated on the database, and the results are projected onto the return expression.

2.2 Extracting PTQL queries

From all events in the training data, we searched for the shortest link paths that connected event triggers to themes, themes to sites, themes to locations, and so on. For each of the different event classes, we obtained a set of link paths connecting the event trigger to the theme. Links from themes to sites (required for phosphorylation, binding, and regulation events) where extracted from all three and then joined into one set. We transformed all linkages into PTQL queries, and ran these queries on the development and test data sets, respectively. Note that this entire process is performed automatically. As many link paths are identical expect for their event trigger terms, we manually grouped similar terms together; queries were then expanded automatically to allow for either one. An example is the following group of inter-changeable terms that could replace 'expression' in gene expression events (see Figure 3):

expression $\equiv$ {*expression, overexpression, coexpression, production, overproduction, generation, synthesis, biosynthesis, transfection, cotransfection*}

For evaluation on the development data, we extracted all queries from the training data; for evaluation on the test set, queries originate from training and development data together.

2.3 Regular expressions for *regulation* events

Regarding regulation events, we concentrated on the recognition of events with only the theme slot filled. In the training data, 73.8% of the regulations (incl. positive and negative regulation) do not have any site, cause, or cause-site arguments/participants. We addressed this task using regular expressions that were matched against the annotated sentences in the PTDB. Therefore, we sought for trigger expressions of regulation events that immediately precede or follow an annotation (protein name or event trigger). For all four possible combinations (precede/follow and protein/trigger) we created regular expressions that were able to recognize the given patterns, for example:

- `(NOUN:trigger) (of) (PROTEIN)`, finds *[up-regulation]*$_{\text{Trigger:Pos\_reg}}$ *of [Fas ligand]*$_{\text{Protein}}$
- `(PROTEIN) (NOUN:trigger)`, finds *mediate [IL-8]*$_{\text{Protein}}$ *[induction]*$_{\text{Trigger:Pos\_reg}}$
- `(VERB:trigger) (EVENT:trigger)`, finds *[inhibit]*$_{\text{Trigger:Neg\_reg}}$ *[secretion]*$_{\text{Event:Loc}}$
- `(EVENT:trigger) (VERB:trigger)`, finds *TNF-alpha [release]*$_{\text{Event:Loc}}$ *[peaked]*$_{\text{Trigger:Pos\_reg}}$

The actual patterns also allowed some event class specific prepositions (of, with, to, etc.) and determiners between the regulation trigger and the protein or event trigger. However, care has to be taken as regulation events often are embedded in nested structures which are not properly recognized by regular expressions. Therefore, whenever a regulation event pattern had been identified, we also constructed another event candidate with the appropriate subexpression as the trigger, such as:

[[IkappaBalpha]$_{\text{Protein}}$ *induction]]*$_{\text{Event:Pos\_reg}}$ *was completely [inhibited]*$_{\text{Trigger:Neg\_reg}}$.

2.4 Context identification to find negations and speculations

We identified *negative context* of events by simultaneously applying four different methods. In the first three methods, we identified candidate *negation trigger expressions* (NTEs) by means of regular expressions that were created based on the analysis of surface patterns of negation annotation in the training set. The fourth method uses the parse trees

```
//S{ //N[value='expression'](e) -> //PRP[value='of'](a)
     => //?[tag='gene'](t) -> //N[value='gene'](h) }
     :  e !Mp a and a !Jp t and t !CH h  : :  e.value, t.value
```

Figure 3: PTQL query for the extraction of some gene expression event. It searches for a sentence S that contains a noun 'expression', followed by a preposition 'of', which is then followed by a noun phrase (2nd line) that contains a gene name ('//?', any node with tag=gene) and has 'gene' as head noun. The link types are specified in the 3rd line using the variables each node is bound to (e,a,t,h): 'expression' has to be connected to 'of' with an 'Mp' link, the link from 'of' to the head noun has to be 'Jp', and the 'CH' link specifies 'gene' as head noun. The return values of the query are the values of nodes 'e' and 't', which are bound to the event trigger 'expression' and the gene, respectively. This query would return all three event/theme pairs from the phrase in Figure 2.

of sentences including negated event using a set of queries for the identification of candidate NTEs. To fine tune the combined prediction, we used some manually encoded exceptions.

1) NTEs inside the trigger of an event: these expressions are partly or entirely event triggers and usually suggest negative context, such as *inability* and *undetectable*. In the training set, sometimes an NTE indicated negation for some event classes but not for others; we added exceptions to exclude such NTE–event class combinations (e.g., *deficient* with a negative regulation).

2) NTEs immediately preceding an annotation (protein name or event trigger), e.g., *no(t), lack of, minimal, absence of, cannot*, etc.

3) NTEs in the span of all the annotation related to an event (triggers, attributes recursively): these NTEs can span over multiple sentences. Starting with a hand-crafted dictionary of negation context triggers (Solt et al., 2009), we selected those dictionary items that had a positive effect on overall F1-measure.

4) NTEs from parse tree patterns: We identified on the training data parse tree patterns including NTEs (using hand-made NTE dictionary) and protein names or event triggers. Candidate patterns, e.g., `regulate`$*\Rightarrow$`in`$\Rightarrow$`but`$\rightarrow$`not`$\Rightarrow$`in`, were then formulated as queries against the PTDB and filtered via optimization.

We also applied the parse tree based method to identify *speculation context* (details not shown). We observed that some apparently speculative contexts were, to our surprise, considered as facts by the annotators if the pattern occurred in the last sentence of the abstract, such as: *These data suggests...*. To counteract such situations, we developed a pattern-location heuristic by dividing the abstract into title, body, and conclusion part. Frequent speculation candidate patterns were evaluated separately on each part and filtered via optimization.

2.5 Resolving anaphora

Almost 8% of all events in the training set span multiple sentences. Our solution outlined so far works at the sentence level and is therefore unable to correctly recognize such events. To overcome this deficiency, we developed a baseline method for anaphora resolution, which is implemented as a pre–processing step. First, we identified all events spanning multiple sentence in the training set and collected typical anaphora expressions for proteins (e.g., *this gene, these proteins, both factors*). For each anaphora occurrence in development and test sets, we searched the closest preceding protein(s); here we also took into account if the anaphora was singular or plural. We also expected that resolved anaphora would generate additional PTQL queries and would thus improve the overall recall twofold. Unfortunately we could not analyze the results of our resolution approach on the train set (due to lack of time) and could hence not take full advantage of this idea. So far, we only addressed anaphora referring to protein(s). Once an anaphora and its referenced expression(s) were recognized, we effectively duplicated the original sentence with referenced expressions substituting the anaphora; PTQL queries would thus run on the original sentence as well as on the resolved version.

2.6 Handling enumerations

In most cases, PTQL queries were able to correctly recognize events that involve enumerated entities. However, when the enumeration included some special characters (brackets, slashes) or led to incorrect parse trees, our queries were not able to extract all annotated events. We applied post-processing to solve this problem, which was applicable when at least one protein in the enumeration was annotated as a part of an event. Post-processing was based on regular expressions searching for additional proteins occurring in the neighborhood of an initial one, separated from it only by an enumeration separator. If found, the original event was replicated by substituting the original protein with the new ones.

3 Datasets and results

Statistics concerning event classes and number of instances per event class can be found in the overview paper for the shared task, see (Kim et al., 2009). All in all, we extracted 1845 different link paths from the training data (2197 from training plus development) that connect two constituents each (event trigger term to protein, or protein to site, for instance), corresponding to as many PTQL queries. Table 1 shows the number of link paths per event class and argument type. From Table 2, which lists the top query per event class according to support in the training data, it becomes obvious that most events are described in fairly simple ways ("gene *expression*" or "*phosphorylation of* gene"). Adding the development data increased the number of events by 20.8% and the number of unique link paths by 19.1%. This might indicate that adding more data in the future will produce less and less new link paths, but we still observe a decent amount of link paths yet not covered. Per link path type, the increase rate ranged from only 9% (localization: theme to atloc) over 11-15% for basic events (gene-expression or transcription trigger term to theme) to almost 27% (regulation: theme to site).

On the BioNLP'09 Shared Task test set, the method achieved an F1-score of 45.6% for the basic types, 9% on regulation events, with a total of 29.3% for Task 2 (see Table 3). On Task 3, the F1-score was 8.6%. For Task 1, which was handled by us implicitly with Task 2, the F1-score was 32.1%. The combined F1-score for all tasks was 29.6%. Precision was significantly higher than recall in all cases (overall: 60% precision at 20% recall).

Concerning regulation events, since we only aimed to recognize the simplest ones with this method, not surprisingly the recall of the method is very low, but the precision is on par with the ones of other events (for positive and negative regulation). The precision gets diminished because only a partial event was submitted, accounting for a false positive and false negative.

The post-processing improved the F1-score of Task 2 slightly (1.2%) for the first 6 events at 3% better recall and 6% worse precision. For regulation

Event class: arguments	**Unique**	**Total**
Localization: event-theme	120	237
Localization: theme-atloc	39	56
Localization: theme-toloc	28	43
Binding: event-theme	578	996
Binding: theme-site	64	130
Gene expression: event-theme	447	1507
Transcription: event-theme	208	498
Protein catabolism: event-theme	42	98
Phosphorylation: event-theme	59	153
Phosphorylation: theme-site	34	60
Regulation: event-theme	178	267
Regulation: protein-site	11	40
Regulation: event-csite	2	2
Regulation: event-cause	35	54
Sum	1845	4141

Table 1: Number of link paths per event class and pair of arguments (based on the training data). Themes are proteins for the first block of events, and proteins or other events for the three regulation types. atloc: at location, toloc: to location.

Event class	**TP**	**FP**	**FN**	**Rec**	**Prec**	**F1**
Localization	42	28	132	24.14	60.00	34.43
Binding	69	86	280	19.77	44.52	27.38
Gene expr.	373	99	349	51.66	79.03	62.48
Transcription	22	30	105	16.06	42.31	23.28
Protein cat.	7	5	7	50.00	58.33	53.85
Phosphoryl.	31	57	108	22.30	35.23	27.31
Sub-total	544	305	991	35.44	64.08	45.64
Regulation	1	12	291	0.34	7.69	0.66
Positive reg.	70	146	917	7.09	32.41	11.64
Negative reg.	14	14	365	3.69	50.00	6.88
Reg. total	85	172	1573	5.13	33.07	8.88
Task 2 total	629	477	2564	19.70	56.87	29.26
Negation	9	24	218	3.96	27.27	6.92
Speculation	13	33	195	6.25	28.26	10.24
Task 3 total	22	57	413	5.06	27.85	8.56
Overall	710	475	2907	19.63	59.92	**29.57**

Table 3: Official results for the BioNLP'09 Shared Task tasks 2 and 3, approximate span, recursive matching.

Pair	Nodes	Links	Support
Localization ↔ theme	GENE(t1) => localization(e1)	t1→CH→e1	36/237
Binding ↔ theme	GENE(t1) => association(e1)	t1→CH→e1	42/996
Gene expression ↔ theme	GENE(t1) => expression(e1)	t1→CH→e1	347/1507
Transcription ↔ theme	GENE(t1) => gene(a1) => transcription(e1)	t1→CH→a1 and a1→CH→e1	72/498
Protein catab. ↔ theme	proteolysis(e1) => of(a1) => GENE(t1)	e1→M→a1 and a1→J→t1	32/98
Phosphorylation ↔ theme	phosphorylation(e1) => of(a1) => GENE(t1)	e1→M→a1 and a1→J→t1	48/153

Table 2: Queries per argument pair (event↔theme) with the highest support in the training data. All nodes are bound to variables (round brackets) that are use in the links to depict connections between nodes. Note that all event trigger terms are placeholders for alternatives (see text): 'expression' also refers to instances that used the terms 'co-expression', 'synthesis', 'production', etc. GENE: wildcard for any gene name; RES: residue. CH: links head noun to modifying noun; M: connects nouns to post-nominal modifiers; J: connects prepositions to objects.

Method	TP	FP	FN	P	R	F1
<u>I</u>nside trigger	15	8	92	65.2	14.0	23.1
<u>B</u>efore trigger	62	17	45	78.5	57.9	66.7
<u>S</u>pan-based	6	3	101	66.7	5.6	10.3
<u>P</u>arse tree query	4	1	103	60.0	5.6	10.7
$(I \cup B \cup S)$	79	27	28	74.5	73.8	74.2
$(I \cup B \cup S \cup P)$	82	28	25	76.6	74.6	**75.6**
$I \cup B \cup S \cup P$ no F	84	79	23	50.9	75.7	60.9

Table 4: Performance for negation context identification on the *development set*. The last row indicates the importance of fine tuning (*F*): when event class–trigger pair exceptions and NTE exceptions are not applied, the precision decreases considerably with only a small increase in recall. See text for details in each method.

events its impact was higher since for those no BioLG based solution was applied. Its overall effect on Task 2 was almost a 4% improvement in F1-score and recall, at 15% decreased precision.

Identification of negative context

Table 4 shows the effectiveness of each method for the identification of negative context on the development set. Searching for the negation inside the event trigger had little effect on the final results, since a specific word was rarely identified as being the trigger of more than one event classes. The most reliable spot to look for negation was immediately before the term that triggered the event (*lack of expression of . . .*).

Identification of speculation

Table 5 shows the effectiveness of our parse tree based method for the identification of speculation context on the development set. With the use of location-based heuristic we could improve the F1-score of our method by 3%, at 7% better precision and 1% worse recall. The parse tree based method worked significantly better for speculative context than for negation, because speculations are expressed in less multifarious way, and trigger words are more specific for the context.

Method	TP	FP	FN	P	R	F1
w/o location hrst.	53	47	42	53.0	55.8	54.4
with location hrst.	52	34	43	60.5	54.7	**57.5**

Table 5: Performance of parse tree based speculation identification, with or without location heuristics; evaluated on the development set.

3.1 Error analysis

An analysis of false positives (FP) and false negatives (FN) revealed the following main types of errors (in order of decreasing gravity). Our system produced much better precision than recall, which is reflected in dominance of FNs over FPs. Note that, as we used parse trees on training and test data, parse errors result both in incorrect queries and wrongly extracted results. Some of these errors, mainly due to missing or incorrect parse trees or links, could be recovered by the post-processing if the surface patterns were simple.

1. FNs: no corresponding link path query
2. FNs: there exists a corresponding yet slightly different link
3. FNs: query links to a (pre or post) modifier of the gene, but not the actual gene name
4. FNs: query misses one argument
5. FPs: wrong event categorization (mostly gene expression vs. transcription)

6. FNs: unseen event trigger term, location, or site
7. FPs: wrong despite perfect match wrt. a link path from the training data
8. FNs, FPs: incorrect or partial parse tree
9. FNs: problems with anaphora, brackets, or enumerations

We discuss these error classes in more detail. The first problem may be attributed to the small size of the training data, but is also a general property of pattern-based methods in NLP. The second class stems from the current inability of our query language to deal with morpho-syntactical variation in language (see next Section). A large portion (3) of false negatives was due to link paths that went to the gene/theme in the training data, but to the head of a noun phrase that contained a gene/theme in the test data (or vice versa); or the link went to a noun pre-modifier. An example is the following, where the first phrase originates from the training data and *gene* is placeholder for the actual gene/protein name:

"... phosphorylates *gene* ..."

"... phosphorylates *gene* protein ..."

"... phosphorylates X domain of *gene* ..."

In all three cases, there is a link from the verb to its object, but in the lower two examples, that object is 'protein' and 'domain', respectively. Only for a few such cases, all three link paths were contained in the training data.

For 5% of the false positive events (5), we predicted the wrong event class, while all trigger terms/arguments were correct. Half of those were mix-ups of positive regulation, predicted as gene expression; another group has gene expression predicted as localization. 13% of FPs were a result of both: the prediction was part of a corresponding FN (but some argument was missing), and at the same time we predicted the wrong type. For a small fraction (1.5%) of false negative events on the development set, we found a corresponding false positive event where one argument (ToLoc, Cause, Site, Theme2) was missing; 11 of those were binding events (comprising 9% of FNs for binding).

A relatively small portion of false negatives were due to non-existing linkages (8) for a sentence. We stopped parsing after 30sec per sentence; this yields partial linkages in some cases, which we could still use for extraction of link paths (training data) or querying against (test data); sometimes, no linkage was available at all. This timeout also influences the quality of linkages, which result in false positives as well as false negatives.

As for context identification, our approach performed significantly weaker on the test set, since over 70% of negations and speculations were related to regulation events (measured on the joined train and development sets), for which we applied a coarse baseline method, i.e., here a large part of the base events were missing.

4 Related work

We focus our discussion on approaches to information extraction that also use LinkGrammar. Evaluations of other deep parsers for information extraction in the life sciences may, for instance, be found in Miyao et al. (2009) and Pyysalo et al. (2008). Note that most other systems based on deep parsing convert IE into a classification problem, often using some kind of convolution kernels, for example, Kim et al. (2008); instead, we employ a pattern-matching approach where patterns are expressed as queries. A similar approach is described in Fundel et al. (2007), where three rules are defined to extract protein-protein interactions from an aggregated form of dependency graphs. These rules could in fact easily be expressed as queries in our language.

Ding et al. (2003) studied the extraction of protein-protein interactions using the Link Grammar parser. After some manual sentence simplification to increase parsing efficiency, their system assumed an interaction whenever two proteins were connected via a link path; an adjustable threshold allowed to cut-off too long paths. As they used the original version of Link Grammar, Ding et al. argue that adaptations to the biomedical domain would enhance the performance. Pyysalo et al. (2004) extracted interaction subgraphs, spanning all predicates and arguments at the same time, from the Link Grammar linkage of known examples. Failure analysis revealed that 34% of the errors were due to unknown grammatical structures, 26% due to dictionary issues and a further 17% due to unknown words.

An adaption of Link Grammar that handles some of the failure cases is BioLG (Pyysalo et al., 2006). BioLG includes additional morpho-guessing rules, lexicon expansion, and disambiguation using a POS tagger. Adding morpho-guessing rules and using a domain-specific POS tagger for disambiguation resulted in an increase from 74.2 to 76.8% in recall; it also increased parsing efficiency by 45%. Szolovits (2003) adapted the Link Grammar parser by expanding the lexicon with data from UMLS Specialist. This expansion consisted of 200k new entries (including 74k phrases), resulting in a 17% increase in coverage on a corpus of 495k words.

The main differences between the cited previous works and our approach are: *1)* we extract only pairwise subgraphs (e.g., from a trigger term to a single protein) and then attempt to construct events based on such small components; *2)* we consider link types, predicates, prepositions, and other nodes as requirements for a valid linkage with respect to event argument recognition; *3)* we use a query language to query persistently stored parse trees instead of parsing each sentence and then comparing it to known link paths; *4)* we combine subgraph matching with extensive pre- and post-processing rules using regular expressions and other filtering rules.

5 Conclusions

We presented a method for extraction of molecular events from text. We distinguished nine classes of events and identified arguments associated with them. We also characterized each event for either being speculative or negated. The underlying method extracts link paths between all relevant pairs of arguments involved in the event from a Link Grammar parse (BioLG, see Pyysalo et al. (2006)). These link paths connect, for instance, an event trigger term to its theme, or a protein theme to a binding site. We query the graph formed by these linkages using a dedicated query language for parse trees (Tu et al., 2008) which allows us to very quickly implement large sets of rules. We combine queries with extensive pre- and post-processing using a mixture of different techniques. For the BioNLP'09 Shared Task, we focused on all event classes but the three types of regulation. For the other six, we obtain an overall F1-score of 45.6%, for all nine it was 29.3% (task 2). Including speculation and negation (task 3), the overall total on all nine event classes was 29.6%. All in all, we found that link paths connecting constituents of known types (e.g., event trigger term, gene) as extracted from training data yield a precise way for event argument detection. Using a specialized query language on pre-processed data (NER; parsing) greatly enhances the utility of such extracted rules to put together more complex events. Still, our current approach lacks in overall recall (20–52%, depending on event class), often due to slight variations that include, for instance, alternative nodes along a link path that were not observed in training data.

```
            +----Js----+               +--Js-+
     +--Mp--+   +--CH--+        +--Mp--+     |
     |      |   |      |        |      |     |
expression of c-Fos gene   expression of c-Fos
```

Figure 4: Example for alternative structures / optional nodes. In this case, the linkage should reflect the connection from 'expression' to a noun that refers to a gene, independent of its head. The 'Mp' and 'Js' links would be required, the 'CH' link from head to actual gene optional.

Our approach could be improved in various ways. First, we currently extract queries from the training corpus and use them directly as they are. We see that to improve recall, queries need to be generalized further. In previous work (Hakenberg et al., 2008) we showed that such generalized rules may be learned automatically (from much larger corpora), which helped to increase recall considerably at a modest precision penalty. Second, our query language currently performs exact matching, while it would be more advantageous to implement some form of fuzzy semantics, producing a ranked list of hits. This could include wildcards, alternative nodes, alternative sub-paths, optional nodes etc. An example is discussed in Figure 4. Finally, we also believe that it would be rather easy to include more sophisticated ways of performing anaphora resolution to properly address events spanning multiple sentences and referential phrases within sentences.

Acknowledgments

We kindly acknowledge funding by Science Foundation Arizona and Alexander-von-Humboldt Stiftung.

References

Steven Bird, Yi Chen, Susan B. Davidson, Haejoong Lee, Yifeng Zheng. 2006. Designing and Evaluating an XPath Dialect for Linguistic Queries. In: *Proc ICDE*, pp.52, Washington, DC, USA.

Jing Ding, Daniel Berleant, Dan Nettleton, Eve S. Wurtele. 2002. Mining MEDLINE: Abstracts, Sentences, or Phrases? In: *Proc. PSB*, pp. 326–337, Kaua'i, Hawaii, USA.

Jing Ding, Daniel Berleant, Jun Xu, Andy W. Fulmer. 2003. Extracting Biochemical Interactions from MEDLINE Using a Link Grammar Parser. In: *Proc IEEE ICTAI*, pp. 467–471.

Katrin Fundel, Robert Küffner, Ralf Zimmer. 2007. RelEx–relation extraction using dependency parse trees. *Bioinformatics*, 23(3):365–371.

Jörg Hakenberg, Conrad Plake, Loic Royer, Hendrik Strobelt, Ulf Leser, Michael Schroeder. 2008. Gene mention normalization and interaction extraction with context models and sentence motifs. *Genome Biology*, 9(S1):S14.

Seonho Kim, Juntae Yoon, Jihoon Yang. 2008. Kernel approaches for genic interaction extraction. *Bioinformatics*, 24(1):118–126.

Jin-Dong Kim, Tomoko Ohta, Sampo Pyysalo, Yoshinobu Kano, Jun'ichi Tsujii. 2009. Overview of BioNLP'09 Shared Task on Event Extraction. In: *Proc Natural Language Processing in Biomedicine (BioNLP) NAACL 2009 Workshop*, June 4-5, Boulder, CO, USA.

Martin Krallinger, Alfonso Valencia, Lynette Hirschman. 2008. Linking genes to literature: text mining, information extraction, and retrieval applications for biology. *Genome Biol*, 9(Suppl 2):S8.

Lucene, available at http://lucene.apache.org

Yusuke Miyao, Kenji Sagae, Rune Sætre, Takuya Matsuzaki, Jun'ichi Tsujii. 2009. Evaluating contributions of natural language parsers to protein-protein interaction extraction. *Bioinformatics*, 25(3):394–400.

Sampo Pyysalo, Filip Ginter, Tapio Pahikkala, Jorma Boberg, Jouni Järvinen, Tapio Salakoski, Jeppe Koivula. 2004. Analysis of Link Grammar on Biomedical Dependency Corpus Targeted at Protein-Protein Interactions. In: *NLPBA/BioNLP at COLING-2004*, pp. 15–21, Geneva, Switzerland.

Sampo Pyysalo, Tapio Salakoski, Sophie Aubin, Adeline Nazarenko. 2006. Lexical adaptation of link grammar to the biomedical sublanguage: a comparative evaluation of three approaches. *BMC Bioinformatics*, 7(Suppl 2):S2.

Sampo Pyysalo, Antti Airola, Juho Heimonen, Jari Björne, Filip Ginter, Tapio Salakoski Pyysalo. 2008. Comparative analysis of five protein-protein interaction corpora. *BMC Bioinformatics*, 9(Suppl 3):S6.

Daniel Sleator and Davy Temperley. 1993. Parsing English with a Link Grammar. In: *Proc 3rd Int Workshop on Parsing Technologies*, Aug 10-13, Tilburg/NL and Durbuy/B.

Peter Szolovits. 2003. Adding a medical lexicon to an English Parser. In *Proc AMIA*, pp. 639–643, Nov 8-12, Washington DC, USA.

Illés Solt, Domonkos Tikk, Viktor Gál, Zsolt T. Kardkovács. 2009. Semantic classification of diseases in discharge summaries using a context-aware rule based classifier. *JAMIA*, in press.

Luis Tari, Jörg Hakenberg, Graciela Gonzalez, Chitta Baral. Querying a Parse Tree Database of Medline Text to Synthesize User-Specific Biomolecular Networks. In *Proc Pac Symp Biocomput*, 14:87-98.

Phan Huy Tu, Chitta Baral, Yi Chen, and Graciela Gonzalez. 2008. Generalized text extraction from molecular biology text using parse tree database querying. Technical Report TR-08-004, Arizona State University.

XML Path Language, see http://www.w3.org/TR/xpath

Tunable Domain-Independent Event Extraction in the MIRA Framework

Georgi Georgiev[1]
georgi.georgiev@ontotext.com

Kuzman Ganchev[1]
kuzman.ganchev@ontotext.com

Vassil Momtchev[1]
vassil.momtchev@ontotext.com

Deyan Peychev[1]
deyan.peychev@ontotext.com

Preslav Nakov[1]
preslav.nakov@ontotext.com

Angus Roberts[2]
a.roberts@dcs.shef.ac.uk

[1] Ontotext AD, 135 Tsarigradsko Chaussee, Sofia 1784, Bulgaria
[2] The Department of Computer Science, Regent Court 211 Portobello, Sheffield, S1 4DP. UK.

Abstract

We describe the system of the PIKB team for BioNLP'09 Shared Task 1, which targets tunable domain-independent event extraction. Our approach is based on a three-stage classification: (1) trigger word tagging, (2) simple event extraction, and (3) complex event extraction. We use the MIRA framework for all three stages, which allows us to trade precision for increased recall by appropriately changing the loss function during training. We report results for three systems focusing on recall (R = 28.88%), precision (P = 65.58%), and F_1-measure (F_1 = 33.57%), respectively.

1 Introduction

Molecular interactions have been the focus of intensive research in the development of in-silico biology. Recent developments like the *Pathway and Interaction Knowledge Base* (*PIKB*) aim to make available to the user the large semantics of the existing molecular interactions data using massive knowledge syndication. PIKB is part of *LinkedLifeData*[1], a platform for semantic data integration based on RDF[2] syndication and lightweight reasoning.

Our system is based on the MIRA framework where, by appropriately changing the loss function on training, we can achieve any desirable balance between precision and recall. For example, low precision with high recall would be appropriate in a search that aims to identify as many potential candidates as possible to be further examined by the user, while high precision might be essential when adding relations to a knowledge base. Such a tunable system is practical for a variety of important tasks, including but not limited to, populating extracted facts in PIKB and reasoning on top of new and old data.

Our system is based on a three-stage classification process: (1) trigger word tagging using a linear sequence model, (2) simple event extraction, and (3) complex event extraction. In stage (2), we generate relations between a trigger word and one or more proteins, while in stage (3), we look for complex interactions between simple events, trigger words and proteins. We use MIRA for all three stages with a loss function tuned for high recall.

[1] *http://www.linkedlifedata.com*
[2] *http://www.w3.org/RDF/*

2 One-best MIRA and Loss Functions

In what follows, x_i will denote a generic input sentence, and y_i will be the "gold" labeling of x_i. For each pair of a sentence x_i and a labeling y, we compute a vector-valued feature representation $f(x_i, y)$. Given a weight vector w, the dot-product $w \cdot f(x, y)$ ranks the possible labelings y of x; we will denote the top scoring labeling as $y_w(x)$. As with hidden Markov models (Rabiner, 1989), $y_w(x)$ can be computed efficiently for suitable feature functions using dynamic programming.

The learning portion of our method requires finding a weight vector w that scores the correct labeling of the training data higher than any incorrect labeling. We used a one-best version of MIRA (Crammer, 2004; McDonald et al., 2005) to choose w. MIRA is an online learning algorithm that updates the weight vector w for each training sentence x_i according to the following rule:

Proceedings of the Workshop on BioNLP: Shared Task, pages 95–98,
Boulder, Colorado, June 2009. ©2009 Association for Computational Linguistics

$$w_{\text{new}} = \arg\min_{w} \|w - w_{\text{old}}\|$$
$$\text{s.t. } w \cdot f(x_i, y_i) - w \cdot f(x, \hat{y}) \geq L(y_i, \hat{y})$$

where $L(y_i, y)$ is a measure of the loss of using y instead of the correct labeling y_i, and $\hat{y}$ is a shorthand for $y_{w_{\text{old}}}(x_i)$. In case of a single constraint, this program has a closed-form solution. The most straightforward and the most commonly used loss function is the Hamming loss, which sets the loss of labeling y with respect to the gold labeling y_i as the number of training examples where the two labelings disagree. Since Hamming loss is not flexible enough for targeted training towards recall or precision, we use a number of task-specific loss functions (see Sections 3 and 5 for details). We implemented one-best MIRA and the corresponding loss functions in an in-house toolkit called Edlin. Edlin provides general machine learning architecture for linear models and a framework with implementations of popular learning algorithms including Naive Bayes, perceptron, maximum entropy, one-best MIRA, and conditional random fields (CRF) among others.

3 Trigger Word Tagging

The training and the development abstracts were first tokenized and split into sentences using maximum entropy models trained on the Genia[3] corpora. Subsequently, we trained several sequence taggers in order to identify the trigger words in text. All our experiments used the standard BIO encoding (Ramshaw and Marcus, 1995) with different feature sets and learning procedures. We focused on recall since it determines the upper bound on the performance of our final system. In our experiments, we found that simultaneously identifying trigger words and the event types they trigger yielded low recall; thus, we settled on identifying trigger words in text as one kind of entity, regardless of event types.

In our initial experiments, we used a CRF-based sequence tagger (Lafferty et al., 2001), which yielded R=43.51%. We further tried feature induction (McCallum, 2003) and second-order Markov assumptions for the CRF, achieving 44.72% and 49.64% recall, respectively.

[3] *http://www-tsujii.is.s.u-tokyo.ac.jp/GENIA/home/wiki.cgi*

Feature Set	**R**	**P**	**F$_1$**
Baseline (current word)	44.82	2.86	05.38
+ POS & char 3-gram	77.41	27.96	41.09
+ previous POS tag	79.77	29.32	42.88
+ lexicon (final tagger)	**80.44**	29.65	43.33

Table 1: Recall (R), precision (P), and F$_1$-measure for the trigger words tagger (in %s) on the development dataset for different feature sets using MIRA training with false negatives as a loss function.

Feature Sets
entity type of e_1 and e_2
words in e_1 and e_2
word bigrams in e_1 and e_2
POS of e_1 and e_2
words between e_1 and e_2
word bigrams between e_1 and e_2
POS between e_1 and e_2
distance between e_1 and e_2
distance between e_1 and e_2 in the dependency graph
steps in parse tree to get e_1 and e_2 in the same phrase
various combinations of the above features

Table 2: Our feature set for the MIRA classifier that predicts binary relations. Here e_1 and e_2 can be *proteins* and/or *trigger words*.

Subsequently, we settled on using MIRA so that we can trade-off precision for recall. In order to boost recall, we defined the loss function as the number of false negative trigger chunks. Thus, a larger loss update was made whenever the model failed to discover a trigger word, while discovering spurious trigger words was penalized less severely. We experimented with popular feature sets previously used for named entity (McCallum and Li, 2003) and gene (McDonald and Pereira, 2005) recognition including orthographic, part-of-speech (POS), shallow parsing and gazetteers. However, we found that only a small number of them was really helpful; a summary is presented in Table 1. In order to boost recall even further, we prepared a gazetteer of trigger chunks derived from the training data, and we extended it with the corresponding WordNet synsets; we thus achieved 80.44% recall for our final tagger.

4 Event Extraction

The input to our event extraction algorithm is a list of trigger words and a list of genes or gene products (e.g., proteins); the output is a set of relations as defined for Task 1. Our algorithm works in two stages. First, we generate events corresponding to relations between a trigger word and one or more proteins (*simple* events); then we generate events for relations between trigger words, proteins and simple events (*complex* events). The two stages differ only in the input data; thus, below we will describe our system for the first stage only.

For each sentence, we considered all pairs of entities (trigger words and proteins), and we used an unstructured classifier to determine the relationship for a given pair. These relationships encoded both the type of event (e.g., *binding*, *regulation*) and entities' roles in that event (e.g., *theme*, *cause*); there was also a special relationship for unrelated entities. We constructed labeled examples to train a MIRA classifier using the training data provided by the task organizers; n-ary relations were then reconstructed from classifier's predictions. The features we used are summarized in Table 2: they are over the words separating the two entities and their part-of-speech tags. We further used some simple features from syntactic phrases (OpenNLP[4] parser) and dependency parse trees (McDonald et al., 2005), extracted using parsers trained on Genia corpora.

After some initial experiments, we found that our features were not sufficiently rich to allow us to learn the relationships between proteins that are part of the same event: we achieved a very low recall of about 20%. Consequently, we focused on the relationships between a trigger word and a protein. Since the competition stipulated that each trigger could be associated with only one type of event, we first chose the event type for each trigger by selecting the protein-label pair with the highest score. We then fixed the event type for this trigger word, and we discarded all proteins for which our classifier assigned a different event type to the target trigger-protein pair. Finally, we added to our output list all binary relations where the role of the protein was *theme*.

For some event classes – *binding*, *regulation*, *positive regulation* and *negative regulation* – the output of the binary classifier was further transformed so that n-ary relations can be formed. However, the way we did this was somewhat ad-hoc. For *binding* events, we added a 3-ary relation between the trigger, the highest scoring protein, and the second highest scoring protein. For *regulation* events, we added a 3-ary relation between the trigger and every pair of proteins where one was a *theme* and the other one was a *cause*. This aggressive addition of potential matches slightly reduced the overall precision, but helped improve the recall for the final system.

[4]http://opennlp.sourceforge.net

Event Class	**R**	**P**	**F_1**
Localization	10.92	82.61	19.29
Binding	7.20	39.68	12.20
Gene expression	30.47	74.58	43.26
Transcription	10.95	39.47	17.14
Protein catabolism	28.57	57.14	38.10
Phosphorylation	34.07	86.79	48.94
Event Total	21.52	68.68	32.77
Regulation	1.37	26.67	2.61
Positive regulation	1.12	25.58	2.14
Negative regulation	0.26	100.00	0.53
Regulation Total	0.97	27.12	1.87
Overall	10.84	64.13	18.55

Table 3: Our official results: for an erroneous submission.

5 Results and Discussion

Unfortunately, we made an error when making our official submission, which resulted in low scores; Table 3 shows the results for that submission.

The rest of this section describes the results and the implementation for the system we *intended to submit*. All reported results are for exact span matches and were obtained using the online tool provided by the task organizers.

As stated in Section 4, we used a linear model trained using one-best MIRA with ten runs over the data for the event extraction system. We oversampled the unstructured training instances that corresponded to a relation so that they become roughly equal in number to those that do not correspond to a relation. Finally, we performed parameter averaging as described in (Freund and Schapire, 1999). These details turned out to be very important for the system performance.

Table 4 shows the results for three different loss functions that gave the best results in our experiments. In describing the loss functions, we define three different types of errors: (1) if the system correctly predicted that a relation should be present,

	0-1 Loss			**High Recall**			**High Precision**		
Event Class	R	P	**F_1**	**R**	P	F_1	R	**P**	F_1
Localization	33.33	69.05	44.96	39.08	48.23	43.17	25.86	86.54	39.82
Binding	38.33	32.60	35.23	46.97	24.51	32.21	24.50	37.95	29.77
Gene expression	57.89	65.72	61.56	64.82	53.49	58.61	47.65	76.27	58.65
Transcription	30.66	33.87	32.18	33.58	22.12	26.67	21.17	47.54	29.29
Protein catabolism	42.86	85.71	57.14	42.86	60.00	50.00	42.86	85.71	57.14
Phosphorylation	75.56	77.86	76.69	77.78	65.22	70.95	52.59	82.56	64.25
Event total	49.64	54.60	52.00	55.98	41.55	47.70	37.93	65.83	48.13
Regulation	0.00	0.00	0.00	2.41	22.58	4.35	0.00	0.00	0.00
Positive regulation	1.73	30.91	3.28	5.29	25.24	8.75	0.20	28.57	0.40
Negative regulation	0.53	40.00	1.04	1.06	23.53	2.02	0.26	100.00	0.53
Regulation Total	1.15	30.16	2.21	3.81	24.80	6.61	0.18	37.50	0.36
Overall	24.45	53.54	**33.57**	**28.88**	39.71	33.44	18.32	**65.58**	28.64

Table 4: Results (in %s) for one-best MIRA with different loss functions.

but guessed the wrong type, we call this a *cross-labeling*; (2) a *false positive* occurs when the learner guessed some relation while there should have been none; (3) the reverse is a *false negative*. All loss functions we considered had a cross-labeling loss of 1. The *0-1 loss* also has a loss of 1 for false positives and false negatives. The *high-recall loss* function penalizes false positives with 0.1 and false negatives with 5. The *high-precision loss* function penalizes false negatives with 0.1 and false positives with 5. The values 0.1 and 5 were chosen on the development data, but were not optimized aggressively.

In conclusion, we have built three domain-independent event extraction systems based on the MIRA framework, each using a different loss function. Overall, they perform quite well and would have been ranked second on precision[5], and 6th on recall, and 7th on F_1-measure.

6 Future Work

After integrating domain knowledge, which should improve the recall for complex events and should boost the overall precision, we intend to transform the system output into RDF and add it to the PIKB repository. The required efforts discouraged us from building a middle ontology between the BioNLP and the PIKB data models, especially given the time limitations for the present task competition. However, we believe this is a promising direction, which we plan to pursue in future work.

[5]Our official submission is second on precision as well.

Acknowledgments

The work reported in this paper was partially supported by the EU FP7 - 215535 LarKC.

References

Koby Crammer. 2004. *Online Learning of Complex Categorial Problems*. Ph.D. thesis, Hebrew University of Jerusalem.

Yoav Freund and Robert E. Schapire. 1999. Large margin classification using the perceptron algorithm. In *Machine Learning*, pages 277–296.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML*. Morgan Kaufmann.

Andrew McCallum and Wei Li. 2003. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In *Proceedings of CoNLL*.

Andrew McCallum. 2003. Efficiently inducing features of conditional random fields. In *Proceedings of UAI*.

Ryan McDonald and Fernando Pereira. 2005. Identifying gene and protein mentions in text using conditional random fields. *BMC Bioinformatics*, (Suppl 1):S6(6).

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of ACL*. ACL.

Lawrence Rabiner. 1989. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2).

Lance Ramshaw and Mitch Marcus. 1995. Text chunking using transformation-based learning. In David Yarovsky and Kenneth Church, editors, *Proceedings of the Third Workshop on Very Large Corpora*. ACL.

BioEve: Bio-Molecular Event Extraction from Text Using Semantic Classification and Dependency Parsing

Syed Toufeeq Ahmed, Radhika Nair, Chintan Patel and Hasan Davulcu
School of Computing and Informatics
Arizona State University
Tempe, Arizona
{toufeeq, ranair1, chpatel, hdavulcu}@asu.edu

Abstract

In this paper, we present **BioEve** a fully automated event extraction system for bio-medical text. It first semantically classifies each sentence to the class type of the event mentioned in the sentence, and then using high coverage hand-crafted rules, it extracts the participants of that event. We participated in Task 1 of BioNLP 2009 Shared task, and the final evaluation results are described here. Our experimentation with different approaches to classify a sentence to bio-interaction classes are also shared.

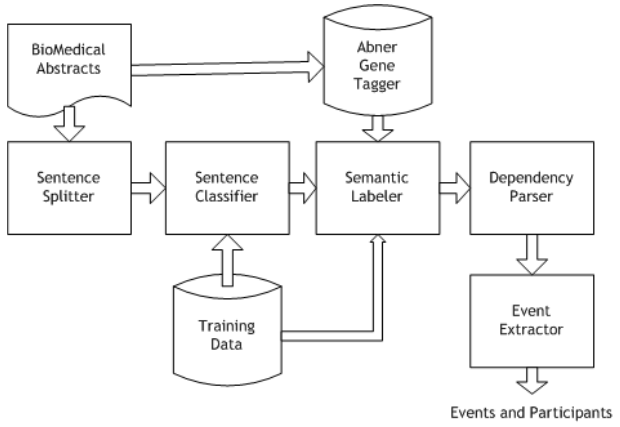


Figure 1: **BioEve** System Architecture

1 Introduction

Human genome sequencing marked beginning of the era of large-scale genomics and proteomics, which in turn led to large amount of information. Lots of that exists (or generated) as unstructured text of published literature. The first step towards extracting event information, in biomedical domain, is to recognize the names of proteins (Fukuda et al., 1998; Blaschke et al., 1999), genes, drugs and other molecules. The next step is to recognize relationship between such entities (Blaschke and Valencia, 2002; Ono et al., 2001; Fundel et al., 2007) and then to recognize the bio-molecular interaction events with these entities as participants (Yakushiji et al., 2001; Tateisi et al., 2004). The BIONLP'09 shared task involved recognition of bio-molecular events, which appear in the GENIA corpus. We mainly focused on task 1, which was detection of an event and its participants.

The rest of the paper is organized as follows. In Section 2 we describe BioEve system, sentence level classification and event extraction using dependency parse tree of the sentence. Sections 3 describes experiments with classification approaches and evaluation results for shared task 1. Section 4 concludes the paper.

2 BioEve: Bio-Molecular Event Extractor

BioEve architecture is shown in Figure 1. First the biomedical abstracts are split into sentences, before being sent to sentence level classifier. We used Näive Bayes Classifier to classify sentences into different event class types. Classification at sentence level is a difficult task, as sentences have lesser information as compared to the whole document. To help event extraction module, each of these sentences are then semantically labeled with additional keywords. We created a dictionary-based labeler, which included trigger words from training data, along with the corresponding event type. These labeled sentences are parsed using a dependency parser to identify `argument-predicate` roles. For each event class type, we hand crafted high coverage extraction rules, similar to Fundel et al. (2007), to identity all event participants. For BioNLP shared task, the event-participant output was formatted to GENIA format.

2.1 Sentence Level Classification and Semantic Labeling

We used Näive Bayes Classifier from Weka [1] library to classify sentences into different event class types. Classification at sentence level is a difficult task, as sentences have lesser information as compared to the whole document. We tried different approaches for classification : 1) Näive Bayes Classifier using bag-of-words, 2) Näive Bayes Classifier using bag-of-words and parts-of-speech tags and 3) SVM Classifier for Weka library.

BioEve event extraction module depends on class labels for extraction. To help with this task, we needed to improve sentence labeling with correct class type information. For this, we employed dictionary based semantic class labeling by identifying trigger (or interaction) words, which clearly indicate presence of a particular event. We used ABNER [2] gene name recognizer to enrich the sentences with gene mentions.

There have been cases in the training data where the same trigger word is associated with more than one event type. To resolve such cases, the trigger words were mapped to the most likely event type based on their occurrence count in the training data. We labeled trigger words in each sentence with their most likely event type. These tagged words served as a starting point for the extraction of event participants. This was done to speed-up the extraction process, as event extraction module now only needs to focus on the parts of the sentences related to these tagged trigger words.

[1]http://www.cs.waikato.ac.nz/ml/weka/

[2]http://pages.cs.wisc.edu/ bsettles/abner/

2.2 Event Extraction Using Dependency Parsing

The sentences, after being class labeled and tagged, are parsed using a dependency parser (Stanford parser[3]) to identify `argument-predicate` roles. Words in the sentence and the relationships between these words form the dependency parse tree of the sentence. For our system, we used typed-dependency representation output format from Stanford parser which is a simple tuple, `reln(gov, dep)`, where `reln` is the dependency relation, `gov` is the governor word and `dep` is the dependent word. Consider the following example sentence:

```
We investigated whether PU.1 binds
and activates the M-CSF receptor
promoter.
```

After this sentence is class labeled and tagged:

```
We investigated whether
T7 binds/BINDING and
activates/POSITIVE_REGULATION the
T8 promoter.
```

The tagged sentence is parsed to obtain dependency relations as shown below:

```
nsubj(investigated-2, We-1)
complm(binds-5, whether-3)
nsubj(binds-5, T7-4)
ccomp(investigated-2, binds-5)
conj_and(binds-5, activates-7)
det(promoter-10, the-8)
nn(promoter-10, T8-9)
dobj(binds-5, promoter-10)
```

This sentence mentions two separate events, *binding* and *positive regulation*. Let's consider the extracting the event *binding* and its participants. Figure 2 shows the parse tree representation and the part of the tree that needs to be identified for extracting event *binding*.

For each event class type, we carefully hand crafted rules, keeping theme of the event, number of participants, and their interactions into consideration. Table 1 lists these extraction rules. In an extraction rule, `T` represents the occurrence of protein in sentence. If multiple proteins are involved, then subscripts, $\mathtt{T}_n$, are used to represent this. The rule

[3]http://nlp.stanford.edu/software/lex-parser.shtml

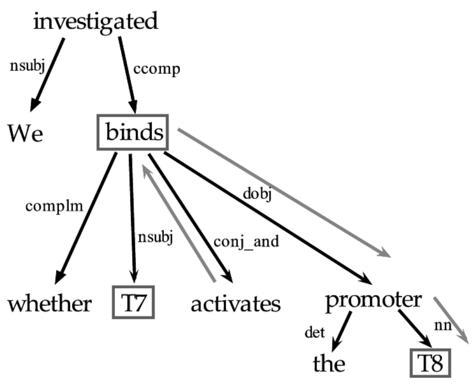


Figure 2: Dependency Parse tree, and event "binding" and its participants are shown.

is triggered when it matches I (for an *interaction word*, or *trigger word*) in the sentence. Some dependency relations and rule predicates are explained below:

- obj(verb/I, T) :- The matching protein is a direct object of the interaction word
- prep(I, T) :- The matching protein is connected to its interaction word by a preposition
- T_1 (I) T_2 : − The interaction word occurs in between the two matching interacting proteins
- conj(T_1, T_2) The two matching proteins are be connected to each other using conjugates such as *'and'*
- ConnectedRule :- The interaction word and the matching protein should be directly connected with a single edge (dependency relation)
- NearestRule :- The interaction word and the matching protein should be connected to each other, directly or indirectly within 5 edge hops, in either direction

Algorithm 1 shows the steps to extract event participants using the rules given in Table 1.

```
Input: Abstract tagged with interaction words
       and class labels
Output: Bio Events with interaction words and
        the participants
foreach abstract do Iterate over each abstract
    foreach sentence in current abstract do
        retrieve all the interaction words in
        current sentence;
        sort them according to precedence of the
        event class type;
        foreach interaction word in the sentence
        do
            extract the participants by matching
            the corresponding event's rule to the
            sentence's dependency parse;
        end
    end
end
```

Algorithm 1: BioEve Event Extraction algorithm

3 Experiments and Evaluations

BioEve shared task evaluation results for Task 1 are shown in Table 2. Event extraction for classes *gene-expression, protein-catabolism and phosphorylation* performed better comparatively, where as, for classes *transcription, regulation, positive-regulation and negative-regulation*, it was below par. The reason noticed (in training examples) was that, most of the true example sentences of *positive-regulation or negative-regulation* class type were mis-classified as either *phosphorylation or gene-expression.* This calls for further improvement of sentence classifier accuracy. Experiments with different approaches for sentence level classification are shown in Table 3. Classifiers were trained on training data and tested on development data. Interestingly, simple Näive Bayes Classifier (NBC) (using just bag-of-words (BOW)) showed better results (up to 10% better) compared to other approaches, even SVM classifier.

4 Conclusions

In this paper, **BioEve**'s Task 1 evaluation results were described, with additional results from different approaches experimented to semantically classify a sentence to the event type. Event extraction performed better for some categories, but clearly needs re-compiling extraction rules for some. Where as classification results showed simple Näive Bayes Classifier performing better than other approaches.

Event Class	Extraction Rules	Event Class	Extraction Rules
Positive Regulation	a) obj(verb/I, T) b) prep(I, T) c) ConnectedRule d) NearestRule	Negative Regulation	a) obj(verb/I, T) b) prep(I, T) c) ConnectedRule d) NearestRule
Regulation	a) prep(I, T) b) ConnectedRule c) NearestRule	Binding	a) T_1 (I) T_2 b) prep(I, T_1); prep(T_1, T_2) c) prep(I, T_1); conj(T_1, T_2) d) obj(verb/I, T) e) prep(I, T) f) ConnectedRule g) NearestRule
Phosphorylation	a) prep(I, T) b) T (connecting-word) I c) ConnectedRule d) NearestRule		
Gene Expression	a) ConnectedRule b) NearestRule	Protein Catabolism	a) prep(I, T) b) ConnectedRule c) NearestRule
Transcription	a) prep(I, T) b) T (connecting-word) I c) ConnectedRule d) NearestRule	Localization	a) prep(I, T) b) ConnectedRule c) NearestRule

Table 1: Extraction rules for each class type. Rules are fired in the order they are listed for each class.

Approach	recall	precision	f-score
Localization	27.59	33.57	30.28
Binding	16.71	30.53	21.60
Gene-expression	**44.04**	**39.55**	**41.68**
Transcription	10.95	11.28	11.11
Prot-catabolism	**57.14**	27.59	37.21
Phosphorylation	**50.37**	**63.55**	**56.20**
Regulation	9.28	5.18	6.65
Pos-regulation	10.48	7.34	8.63
Neg-regulation	12.93	10.19	11.40
All Total	**21.81**	**18.21**	**19.85**

Table 2: BioNLP Shared Task Evaluation: Task 1 Results using approximate span matching.

Sentence Classifier	Correct	Incorrect
NBC(BOW)	**60.45%**	**39.54%**
NBC(BOW+POS)	43.12%	56.87%
SVM	50.14%	49.85%

Table 3: Sentence Classifier results for different approaches: 1) Näive Bayes Classifier (NBC) (using bag-of-words (BOW)), 2) Näive Bayes Classifier(using BOW + Parts-of-speech(POS) tags) and 3) SVM Classifier. Total number of instances =**708**.

References

Christian Blaschke and Alfonso Valencia. 2002. The frame-based module of the suiseki information extraction system. *IEEE Intelligent Systems*, 17(2):14–20.

C. Blaschke, MA. Andrade, C. Ouzounis, and A. Valencia. 1999. Automatic extraction of biological information from scientific text: protein-protein interaction. In *Proceedings of the AAAI conference on Intelligent Systems in Molecular Biology*, pages 60–7. AAAI.

K. Fukuda, A. Tamura, T. Tsunoda, and T. Takagi. 1998. Toward information extraction: identifying protein names from biological papers. In *Pac Symp Biocomput*, volume 707, page 18.

Katrin Fundel, Robert Küffner, and Ralf Zimmer. 2007. Relex—relation extraction using dependency parse trees. *Bioinformatics*, 23(3):365–371.

Toshihide Ono, Haretsugu Hishigaki, Akira Tanigami, and Toshihisa Takagi. 2001. Automated extraction of information on protein-protein interactions from the biological literature. *Bioinformatics*, 17(2):155–161.

Y. Tateisi, T. Ohta, and J. Tsujii. 2004. Annotation of predicate-argument structure of molecular biology text. In *JCNLP-04 workshop on Beyond Shallow Analyses*.

Akane Yakushiji, Yuka Tateisi, Yusuke Miyao, and Jun ichi Tsujii. 2001. Event extraction from biomedical papers using a full parser. In *Pac. Symp. Biocomput*, pages 408–419.

From Protein-Protein Interaction to Molecular Event Extraction

Rune Sætre[†], **Makoto Miwa**[†], **Kazuhiro Yoshida**[‡] and **Jun'ichi Tsujii**[†]
{rune.saetre,mmiwa,kyoshida,tsujii}@is.s.u-tokyo.ac.jp
[†]Department of Computer Science
[‡]Information Technology Center
University of Tokyo
Hongo 7-3-1, Bunkyo-ku, Tokyo, Japan

Abstract

This document describes the methods and results for our participation in the BioNLP'09 Shared Task #1 on Event Extraction. It also contains some error analysis and a brief discussion of the results. Previous shared tasks in the BioNLP community have focused on extracting *gene and protein names*, and on finding (direct) protein-protein interactions (PPI). This year's task was slightly different, since the protein names were already manually annotated in the text. The new challenge was to extract biological *events* involving these given *gene and gene products*. We modified a publicly available system (AkanePPI) to apply it to this new, but similar, protein interaction task. AkanePPI has previously achieved state-of-the-art performance on all existing public PPI corpora, and only small changes were needed to achieve competitive results on this event extraction task. Our official result was an F-score of 36.9%, which was ranked as number six among submissions from 24 different groups. We later balanced the recall/precision by including more predictions than just the most confident one in ambiguous cases, and this raised the F-score on the test-set to 42.6%. The new Akane program can be used freely for academic purposes.

1 Introduction

With the increasing number of publications reporting on protein interactions, there is also a steadily increasing interest in extracting information from Biomedical articles by using Natural Language Processing (BioNLP). There has been several *shared tasks* arranged by the BioNLP community to compare different ways of doing such Information Extraction (IE), as reviewed in Krallinger et al.(2008).

Earlier shared tasks have dealt with Protein-Protein Interaction (PPI) in general, but this task focuses on more specific molecular events, such as *Gene_expression, Transcription, Protein_catabolism, Localization and Binding*, plus *(Positive or Negative) Regulation* of proteins or other events. Most of these events are related to PPI, so our hypothesis was that one of the best performing PPI systems would perform well also on this new event extraction task. We decided to modify a publicly available system with flexible configuration scripting (Miwa et al., 2008). Some adjustments had to be made to the existing system, like adding new types of Named Entities (NE) to represent the *events* mentioned above. The modified AkaneRE (for Relation Extraction) can be freely used in academia[1].

2 Material and Methods

The event extraction system is implemented in a pipeline fashion (Fig. 1).

2.1 Tokenization and Sentence Boundary Detection

The text was split into single sentences by a simple sentence detection program, and then each sentence was split into words (tokens). The tokenization was done by using white-space as the token-separator, but since all protein names are known during both training and testing, some extra tokenization rules were applied. For example, the protein

[1]http://www-tsujii.is.s.u-tokyo.ac.jp/∼satre/akane/

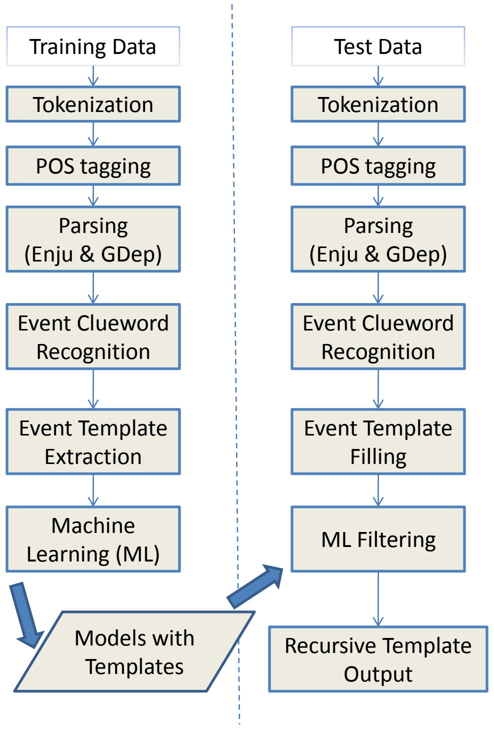


Figure 1: System Overview

name “T cell factor 1” is treated as a single token, “T_cell_factor_1”, and composite tokens including a protein name, like “(T_cell_factor_1)”, are split into several tokens, like ’(’, ’T_cell_factor_1’ and ’)’, by adding space around all given protein names. Also, punctuation (*commas*, *periods* etc.) were treated as separate tokens.

2.2 POS-tagging and Parsing

We used Enju[2] and GDep[3] to parse the text. These parsers have their own built-in Part-of-Speech (POS) taggers, and Enju also provides a normalized lemma form for each token.

2.3 Event Clue-word tagging

Event clue-word detection was performed by a Machine Learning (ML) sequence labeling program. This named-entity tagger program is based on a first order Maximum Entropy Markov Model (MEMM) and is described in Yoshida and Tsujii (2007). The clue-word annotation of the shared-task training set was converted into BIO format, and used to train the MEMM model. The features used in the MEMM model was extracted from surface strings and POS information of the words corresponding to (or adjacent to) the target BIO tags. The clue-word tagger was applied to the development and test sets to obtain the marginal probability that each word is a clue-word of a certain category. The probabilities were obtained by marginalizing the n-best output of the MEMM tagger. We later also created clue-word probability annotation of the training set, to enable the template extraction program to access clue-word probability information in the training phase.

2.4 Event Template Extraction

The training data was used to determine which events to extract. As input to the system, a list of *Named Entity* (NE) types and the *Roles* they can play were provided. The roles can be thought of as slots for arguments in event-frames, and in this task the roles were *Event (clue)*, *Theme* and *Cause*. In the original AkanePPI (based on the AIMed corpus), the only NE type was *Protein*, and the only role was *Theme* (p1 and p2). All the (PPI) events were pairwise *interactions*, and there was no explicit *event-clue* role. This means that all the events could be represented with the single template shown first in Table 1.

The BioNLP shared task used eight other NE types, in addition to manually annotated *Proteins*, namely *Binding*, *Gene_expression*, *Localization*, *Protein_catabolism*, *Transcription*, *Regulation*, *Positive_Regulation* and *Negative_Regulation*. The first five events have only *Theme* slots, which can only be filled by *Proteins*, while the last three regulation events are very diverse. They also have one *Theme* slot, but they can have a *Cause* slot as well, and each role/slot can be filled with either *Proteins*, or other *Events*. See the first half of Table 1.

148 templates were extracted and clustered into nine homogeneous groups which were classified as nine separate sub-problems. The grouping was based on whether the templates had an *Event* or a *Protein* in the same role-positions. This way of organizing the groups was motivated by the fact that the *Proteins* are 100% certain, while the accuracy of the clue-word recognizer is only around 50% (estimated on the training data). The bottom of Table 1 shows the resulting nine **general** interaction templates.

[2]http://www-tsujii.is.s.u-tokyo.ac.jp/enju/
[3]http://www.cs.cmu.edu/∼sagae/parser/gdep/

2.5 Machine Learning with Maximum Entropy Models

We integrated Maximum Entropy (ME) modeling, also known as Logistic Regression, into AkaneRE. This was done by using LIBLINEAR[4], which handles multi-class learning and prediction. Gold templates were extracted during training, and each template was matched with all legal combinations of Named Entities (including gold proteins/clue-words and other recognized clue-word candidates) in each sentence. The positive training examples were labeled as gold members of the template, and all other combinations matching a given template were labeled as negative examples within that specific template class. The templates were grouped into the nine *general* templates shown in the bottom of Table 1. Using one-vs-rest logistic regression, we trained one multi-class classifier for each of the nine groups individually. The ML features are shown in Table 2.

In the test-phase, we extracted and labeled all relation candidates matching all the templates from the training-phase. The ML component was automatically run independently for each of the nine groups listed in the bottom of Table 1. Each time, all the candidate template-instances in the current group were assigned a confidence score by the classifier for that group. This score is the probability that a candidate is a true relation, and a value above a certain threshold means that the extracted relation will be predicted as a true member of its specific template. LIBLINEAR's C-value parameter and the prediction threshold were selected by hand to produce a good F-score (according to the strict matching criterion) on the development-test set.

2.6 Filtering and recursive output of the most confident template instances

After machine learning, all the template instances were filtered based on their confidence score. After tuning the threshold to the development test-set, we ended up using 1 as our C-value, and 3.5% as our confidence threshold. Because the prediction of *Regulation Events* were done independent from the sub-events (or proteins) affected by that event, some sub-events had to be included for completeness, even if their confidence score was below the threshold.

[4]http://www.csie.ntu.edu.tw/∼cjlin/liblinear/

3 Results and Discussion

Our final official result was an F-score of 36.9%, which was ranked as number six among the submissions from 24 different groups. This means that the AkanePPI system can achieve good results when used on other PPI-related relation-extraction tasks, such as this first BioNLP event recognition shared task. The most common error was in predicting regulation events with other events as *Theme or Cause*. The problem is that these events involve more than one occurrence of event-trigger words, so the performance is more negatively affected by our imperfect clue-word detection system.

Since the recall was much lower on the test-set than on the development test-set, we later allowed the system to predict multiple confident alternatives for a single event-word, and this raised our score on the test-set from 36.9% to 42.6%. In hindsight, this is obvious since there are many such examples in the training data: E.g. "over-express" is both positive_regulation and Gene_expression. The new system, named AkaneRE (for Relation Extraction), can be used freely for academic purposes.

As future work, we believe a closer integration between the clue-word recognition and the template prediction modules can lead to better performance.

Acknowledgments

"Grant-in-Aid for Specially Promoted Research" and "Genome Network Project", MEXT, Japan.

References

Martin Krallinger et al. 2008. Evaluation of text-mining systems for biology: overview of the second biocreative community challenge. *Genome Biology*, 9(S2).

Makoto Miwa, Rune Sætre, Yusuke Miyao, Tomoko Ohta, and Jun'ichi Tsujii. 2008. Combining multiple layers of syntactic information for protein-protein interaction extraction. In *Proceedings of SMBM 2008*, pages 101–108, Turku, Finland, September.

Kazuhiro Yoshida and Jun'ichi Tsujii. 2007. Reranking for biomedical named-entity recognition. In *Proceedings of the Workshop on BioNLP 2007*, June. Prague, Czech Republic.

Freq	**Event**	**Theme1**	**Theme2**	**Theme3**	**Theme4**	**Cause**
-	PPI	Protein	Protein			
613	Binding	Protein				
213	Binding	Protein	Protein			
3	Binding	Protein	Protein	Protein		
2	Binding	Protein	Protein	Protein	Protein	
217	Regulation	Protein				Protein
12	Regulation	Binding				Protein
48	+Regulation	Transcription				Protein
4	+Regulation	Phosphorylation				Binding
5	-Regulation	+Regulation				Protein
...	...	...				...
Total	148 Templates					
Count	General Templates	**Theme1**	**Theme2**	**Theme3**	**Theme4**	**Cause**
9	event templates	Protein				
1	event template	Protein	Protein			
1	event template	Protein	Protein	Protein		
1	event template	Protein	Protein	Protein	Protein	
3	event templates	Protein				Protein
12	event templates	Protein				Event
27	event templates	Event				
26	event templates	Event				Protein
68	event templates	Event				Event

Table 1: Interaction Templates from the training-set. Classic PPI at the top, compared to Binding and Regulation events in the middle. 148 different templates were automatically extracted from the training data by AkaneRE. At the bottom, the Generalized Interaction Templates are shown, with proteins distinguished from other Named Entities (Events)

Feature	Example
Text	The **binding** of the most prominent factor, named TCF-1 (**T_cell_factor_1**), is correlated with the proto-enhancer activity of TCEd.
BOW_B BOW_M0 BOW_A	The -comma- -lparen- factor most named of prominent PROTEIN the -comma- -rparen- activity correlated is of proto-enhancer the TCEd with
Enju_PATH pairs triples	(**ENTITY1**) (<prep_arg12arg1) (**of**) (prep_arg12arg2>) (**factor**) (<verb_arg123arg2) (**name**) (verb_arg123arg3>) (**ENTITY2**) (**ENTITY1** <prep_arg12arg1) (<prep_arg12arg1 **of**) (**of** prep_arg12arg2>) ... (**ENTITY1** <prep_arg12arg1 **of**) (<prep_arg12arg1 **of** prep_arg12arg2>) ...
GDep_PATH pairs/triples	(**ENTITY1**) (<NMOD) (**name**) (<VMOD) (**ENTITY2**) (**ENTITY1** <NMOD) (<NMOD **name**) ... (**ENTITY1** <NMOD **name**) ...
Vector	BOW_B BOW_M0...BOW_M4 BOW_A Enju_PATH GDep_PATH

Table 2: Bag-Of-Words (BOW) and shortest-path features for the machine learning. Several BOW feature groups were created for each template, based on the position of the words in the sentence, relative to the position of the template's Named Entities (NE). Specifically, BOW_B was made by the words from the beginning of the sentence to the first NE, BOW_A by the words between the last NE and the end of the sentence, and BOW_M0 to BOW_M4 was made by the words between the main event clue-word and the NE in slot 0 through 4 respectively. The path features are made from one, two or three neighbor nodes. We also included certain specific words, like "binding", as features.

A Multi-Phase Approach to Biomedical Event Extraction

Hyoung-Gyu Lee, Han-Cheol Cho, Min-Jeong Kim
Joo-Young Lee, Gumwon Hong, Hae-Chang Rim
Department of Computer and Radio Communications Engineering
Korea University
Seoul, South Korea
{hglee,hccho,mjkim,jylee,gwhong,rim}@nlp.korea.ac.kr

Abstract

In this paper, we propose a system for biomedical event extraction using multi-phase approach. It consists of event trigger detector, event type classifier, and relation recognizer and event compositor. The system firstly identifies triggers in a given sentence. Then, it classifies the triggers into one of nine predefined classes. Lastly, the system examines each trigger whether it has a relation with participant candidates, and composites events with the extracted relations. The official score of the proposed system recorded 61.65 precision, 9.40 recall and 16.31 f-score in approximate span matching. However, we found that the threshold tuning for the third phase had negative effect. Without the threshold tuning, the system showed 55.32 precision, 16.18 recall and 25.04 f-score.

1 Introduction

As the volume of biomedical literature grows exponentially, new biomedical terms and their relations are also generated. However, it is still not easy for researchers to access necessary information quickly since it is lost within large volumes of text. This is the reason that the study of information extraction is receiving the attention of biomedical and natural language processing (NLP) researchers today.

In the shared task, the organizers provide participants with raw biomedical text, tagged biomedical terms (proteins), and the analyzed data with various NLP techniques such as tokenization, POS-tagging, phrase structure and dependency parsing and so on. The expected results are the events, which exist in the given text, consisting of a trigger and its participant(s) (Kim et al., 2009).

The proposed system consists of three phases; event trigger detection phase(TD phase), event type classification phase(TC phase), relation recognition and event composition phase(RE phase). It works in the following manner. Firstly, it identifies triggers of a given biomedical sentence. Then, it classifies triggers into nine pre-defined classes. Lastly, the system finds the relations between triggers and participant candidates by examining each trigger whether it has relations with participant candidates, and composites events with the extracted relations. In the last phase, multiple relations of the same trigger can be combined into an event for *Binding* event type. In addition, multiple relations can be combined and their participant types can be classified into not only *theme* but also *cause* for three *Regulation* event types.

In this paper, we mainly use dependency parsing information of the analyzed data because several previous studies for SRL have improved their performance by using features extracted from this information (Hacioglu, 2004; Tsai et al., 2006).

In the experimental results, the proposed system showed 68.46 f-score in TD phase, 85.20 accuracy in TC phase, 89.91 f-score in the initial step of RE phase and 81.24 f-score in the iterative step of RE phase, but officially achieved 61.65 precision, 9.40 recall and 16.31 f-score in approximate span matching. These figures were the lowest among twenty-four shared-task participants. However, we found that the threshold tuning for RE phase had caused a negative effect. It deteriorates the f-score of the

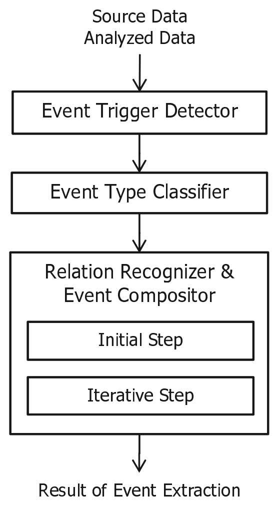


Figure 1: System Architecture

proposed system by enlarging the gap between precision and recall. With the default threshold, the system showed better result in the final test data, 55.32 precision, 16.18 recall and 25.04 f-score with the rank 17th among 24 teams.

2 System Description

Figure 1 shows our bio-event extraction system which consists of Event Trigger Detector, Event Type Classifier and Relation Recognizer & Event Compositor. Each component includes single or multiple Maximum Entropy models trained by gold annotation data. The inputs of the system are source data and analyzed data. The former is raw text with entity annotation, and the latter is tokenized, POS tagged and parsed data of the raw text.[1]

Because the event type is useful to recognize the relation, we perform TC phase before RE phase.

One of important characteristics of bio-event is that one event as well as a protein may participate in another event. Considering this, we designed the system in which the Relation Recognizer be performed through two steps. In the initial step, the systems examines each trigger whether it has the relations with only proteins, and composites events with recognized relations. In the iterative step, it repeatedly examines remained triggers in the same manner. This step allows the system to extract chain-style events, which means that one event participates in another one and the other participates in the former.

To increase the f-score, we tuned a threshold for RE phase which is a binary classification task; deciding whether a given relation candidate is correct one or not. When the output probability of a maximum entropy model is lower than the threshold, we discard a relation candidate.

2.1 Event Trigger Detection

We assume that an event trigger is a single word. In other words, we do not consider the multi-word trigger detection. Because the trigger statistic in the training data showed that about 93% of triggers are single word, we concentrated on the single word trigger detection.

This phase is simply defined as the task that classify whether each token is a trigger or not in a document. It is necessary to select targets to classify among all tokens, because a set of all tokens includes too many negative examples. For this, the following filtering rules are applied to each token. Though these rules filtered out 69.5% of tokens, the trigger recall was 94.8%.

- Filter out tokens whose POS tag is not matched to anything among NN, NNS, VB, VBD, VBG, VBN, VBP, VBZ, JJ and JJR.
- Filter out tokens that are a biomedical named entity.
- Filter out sentences that do not have any proteins.

Proposed features for the binary classification of tokens include both features similar to those used in (Hacioglu, 2004; Tsai et al., 2006; Ahn, 2006) and novel ones. The selected feature set is showed in Table 1.

2.2 Event Type Classification

In TC phase, tokens recognized as trigger are classified into nine pre-defined classes. Although more than a dozen features had been tested, the features except word and lemma features hardly contributed to the performance improvement. The tuned feature set is showed in Table 2.

[1]We used the *GDep* result provided by organizers of the shared task as analyzed data.

Word level features
- Token word
- Token lemma
- Token POS
- POSs of previous two tokens
- Distance, word and POS of the nearest protein
- Positional independence: Whether a noun or a verb is adjacent to the current token
Dependency level features
- Dependency label path of the nearest protein
- The existence of protein in family: This feature is motivated by the study in (Hacioglu, 2004)
- A boolean feature which is true if token's child is a proposition and the chunk of the child include a protein
- A boolean feature which is true if token's child is a protein and its dependency label is OBJ

Table 1: Features for event trigger detection

Features for the event type classification
- Trigger word
- Trigger lemma
- A boolean feature which is true if a protein exists within left and right two words

Table 2: Features for event type classification

We found that TC phase showed relatively high precision and recall with simple lexical features in the experiment. However, it was quite difficult to find additional features that could improve the performance.

2.3 Relation Recognition and Event Composition

In the last phase, the system examines each trigger whether it has relations with participant candidates, and composites events with the extracted relations. (A relation consists of one trigger and one participant)

We devised a two-step process, consisting of initial and iterative steps, because a participant candidate can be a protein or an event. In the initial step, the system finds relations between triggers and protein participant candidates. Features are explained in Table 3. Then, it generates one event with one relation for event types that have only one participant. For *Binding* event type, the system combines at most three relations of the same trigger into one event. For *Regulation* event types, we trained a binary classifier to classify participants of a *Regulation* event into *theme* or *cause*. Features for participant type classification is explained in Table 4. Among multiple participants of a *Regulation* event, only two participants having highest probabilities for *theme* and *cause* constitute one event.

In the iterative step, the system finds relations between triggers and event participant candidates that were extracted in the previous step, and generates events in the same manner. The system performs iterative steps three times to find chain events.

Features are basically common in the initial (I-1) step and the iterative (I-2) step, but some features improve the performance only in one step. In order to represent the difference in Table 3, we indicate (I-1) when a feature is used in the initial step only, and indicate (I-2) when it used in the iterative step only.

Word level features
- Trigger word
- Trigger lemma
- Trigger type (I-1)
- Entity word
- Entity type (I-2)
- Word sequence between T&P (I-1)
- Word distance
- Existence of another trigger between T&P
- The number of triggers of above feature
- Existence of another participant candidate
- The number of participants of above feature
Dependency level features
- Trigger dependency label (I-1)
- Entity dependency label
- Lemma of trigger's head word (I-1)
- POS of trigger's head word
- Lemma of entity's head word (I-1)
- POS of entity's head word
- Lemma of trigger's head word + Lemma of entity's head word
- Right lemma of trigger's head word
- 2nd right lemma of trigger's head word (I-1)
- Right lemma of entity's head word
- 2nd right lemma of entity's head word (I-1)
- Dependency path between T&P
- Dependency distance between T&P
- Direct descendant: a participant candidate is a direct descendant of a given trigger

Table 3: Features for relation recognition between a trigger and a participant (T&P)

Word level features
- Trigger word
- Trigger lemma
- Participant words - event's trigger words if a participant is an event
- Left lemma of a participant
- Right lemma of a participant
- Trigger word + Participant words
- Trigger lemma + Participant lemmas
- Participant lemmas
- Right lemma of a trigger
- 2nd right lemma of a trigger
- Right lemma of a participant
- 2nd left lemma of a participant
Dependency level features
- Dependency path
- Dependency relation to trigger's head
- Dependency relation to participant's head
- POS pattern of common head chunk of a trigger and a participant
- POS pattern of common head chunk of a trigger and a participant + The presence of an object word in dependency path

Table 4: Features of the participant type classifier for *Regulation* events

3 Experimental Result

Table 5 shows the official results of the final test data. After the feature selection, we have performed the experiments with the development data to tune the threshold to be used in RE phase. The work improved the performance slightly. The new threshold discovered by the work was 0.65 rather than the default value, 0.5. However, we found that the tuned threshold was over-fitted to development data. When we tested without any threshold change, the proposed system showed better f-score by reducing the gap between precision and recall. Table 6 shows the performance in this case.

Nevertheless, recall is still quite lower than precision in Table 6. The reason is that many triggers are not detected in TD phase. The recall of the trigger detector was 63% with the development data. Analyzing errors of TD phase, we found that the system missed terms such as *role, prevent* while it easily detected bio-terms such as *phosphorylation, regulation*. It implies that the word feature causes not only high precision but also low recall in TD phase.

Event equality	recall	precision	f-score
Strict	8.99	58.97	15.60
Approximate Span	9.40	61.65	16.31

Table 5: The official results with threshold tuning

Event equality	recall	precision	f-score
Strict	15.46	52.85	23.92
Approximate Span	16.18	55.32	25.04

Table 6: The results without threshold tuning

4 Conclusion

In this paper, we have presented a biomedical event extraction system consisting of trigger detector, event type classifier and two-step participant recognizer. The system uses dependency parsing and predicate argument information as main sources for feature extraction.

For future work, we would like to increase the performance of TD phase by adopting two-step method similar to RE phase. We also will exploit more analyzed data such as phrase structure parsing information to improve the performance.

References

Kadri Hacioglu. 2004. *Semantic Role Labeling Using Dependency Trees*. In *Proceedings of COLING-2004*, Geneva, Switzerland.

Richard Tzong-Han Tsai, Wen-Chi Chou, Yu-Chun Lin, Cheng-Lung Sung, Wei Ku, Ying-shan Su, Ting-Yi Sung and Wen-Lian Hsu. 2006. *BIOSMILE: Adapting Semantic Role Labeling for Biomedical Verbs: An Exponential Model Coupled with Automatically Generated Template Features*. In *Proceedings of BioNLP-2006*.

Mihai Surdeanu, Sanda Harabagiu, John Williams and Paul Aarseth. 2003. *Using Predicate-Argument Structures for Information Extraction*. In *Proceedings of ACL-2003*, Sapporo, Japan.

David Ahn. 2006. *The stages of event extraction*. In *Proceedings of Workshop On Annotating And Reasoning About Time And Events*.

Jin-Dong Kim, Tomoko Ohta, Sampo Pyysalo, Yoshinobu Kano and Jun'ichi Tsujii. 2009. *Overview of BioNLP'09 Shared Task on Event Extraction*. In *Proceedings of Natural Language Processing in Biomedicine (BioNLP) NAACL 2009 Workshop*.

Supervised Classification for Extracting Biomedical Events

Arzucan Özgür
Department of EECS
University of Michigan
Ann Arbor, MI 48109, USA
ozgur@umich.edu

Dragomir R. Radev
Department of EECS and
School of Information
University of Michigan
Ann Arbor, MI 48109, USA
radev@umich.edu

Abstract

We introduce a supervised approach for extracting bio-molecular events by using linguistic features that represent the contexts of the candidate event triggers and participants. We use Support Vector Machines as our learning algorithm and train separate models for event types that are described with a single theme participant, multiple theme participants, or a theme and a cause participant. We perform experiments with linear kernel and edit-distance based kernel and report our results on the BioNLP'09 Shared Task test data set.

1 Introduction

Most previous work on biomedical information extraction focuses on identifying relationships among biomedical entities (e.g. protein-protein interactions). Unlike relationships, which are in general characterized with a pair of entities, events can be characterized with event types and multiple entities in varying roles. The BioNLP'09 Shared Task addresses the extraction of bio-molecular events from the biomedical literature (Kim et al., 2009). We participated in the "Event Detection and Characterization" task (Task 1). The goal was to recognize the events concerning the given proteins by detecting the event triggers, determining the event types, and identifying the event participants.

In this study, we approach the problem as a supervised classification task. We group the event types into three general classes based on the number and types of participants that they involve. The first class includes the event types that are described with a single theme participant. The second class includes the event types that are described with one or more theme participants. The third class includes the events that are described with a theme and/or a cause participant. We learn support vector machine (SVM) models for each class of events to classify each candidate event trigger/participant pair as a real trigger/participant pair or not. We use various types of linguistic features such as lexical, positional, and dependency relation features that represent the contexts of the candidate trigger/participant pairs. The results that we submitted to the shared task were based on using a linear kernel function. In this paper, we also report our results based on using an edit-distance based kernel defined on the shortest dependency relation type paths between a candidate trigger/participant pair.

2 System Description

2.1 Event Type Classes

We grouped the nine event types targeted at the BioNLP'09 Shared Task into three general event classes based on the number and types of participants that they involve.

Class 1 Events: Events that involve a single theme participant (Gene expression, Transcription, Protein catabolism, Localization, and Phosphorylation event types).

Class 2 Events: Events that can involve one or more theme participants (Binding event type).

Class 3 Events: Events that can be described with a theme and/or a cause participant (Regulation, Positive regulation, and Negative regulation event types). Unlike Class 1 and Class 2 events, where the participants are proteins, the participants of Class 3 events can be proteins or events.

Since the event types in each class are similar to each other based on the number and roles of participants that they involve and different from the event types in the other classes, we learned separate classification models for each class. We formulated the classification task as the classification of trigger/participant pairs. We extracted positive and negative training instances (trigger/participant pairs) from the training data for each class of events. We considered only the pairs that appear in the same sentence. We used the tokenized and sentence split abstracts provided by the shared task organizers[1]. Consider the sentence *"The phosphorylation of TRAF2 inhibits binding to the CD40 cytoplasmic domain"*. This sentence describes the following three events:

1. Event1: Type: Phosphorylation Trigger: phosphorylation Theme: TRAF2

2. Event2: Type: Binding Trigger: binding Theme1: TRAF2 Theme2: CD40

3. Event3: Type: Negative regulation Trigger: inhibits Theme: Event2 Cause: Event1

Event1 belongs to Class 1. The trigger/participant pair (phosphorylation, TRAF2) is a positive instance for Class 1. Event2 belongs to Class 2. It has two theme participants. The instances for Class 2 events are created by decomposing the events into trigger/theme pairs. The two positive instances extracted from the decomposition of Event2 are (binding, TRAF2) and (binding, CD40). Event3 belongs to Class 3. It consists of two semantically different participants, namely a theme and a cause. We trained two separate models for Class 3 events, i.e., one model to classify the themes and another model to classify the causes. Another distinguishing characteristic of Class 3 events is that a participant of an event can be a protein or an event. We represent the participants that are events with their corresponding event triggers. We decompose Event3 into its theme and cause and represent its cause Event1 with its trigger word "phosphorylation" and its theme Event2 with its trigger word "binding". As a result, (inhibits, binding) and (inhibits, phosphorylation) are included as positive instances to the Class 3 theme and Class 3 cause training sets, respectively. Negative instances for Class 1 and Class 2 are created by including all the trigger/protein pairs which are not among the positive instances of that class. Negative instances for Class 3 theme and Class 3 cause are created by including all the trigger/protein and trigger1/trigger2 pairs which are not among the positive instances of that class. For example, (phosphorylation, CD40) is a negative instance for Class 1 and (inhibits, TRAF2) is a negative instance for Class 3 theme and Class 3 cause.

[1]http://www-tsujii.is.s.u-tokyo.ac.jp/GENIA /SharedTask/tools.html

2.2 Feature Extraction

2.2.1 Lexical and Part-of-Speech Features

We used the candidate trigger and its part-of-speech, which was obtained by using the Stanford Parser, as features, based on our observation that different candidate triggers might have different likelihoods of being a real trigger for a certain event. For example, *"transcription"* is a trigger for the Transcription event 277 times in the training set and has not been used as a trigger for other types of events. On the other hand, *"concentration"* is used only once as a trigger for a Transcription event and three times as a trigger for Regulation events.

2.2.2 Positional Features

We used two features to represent the relative position of the participant with regard to the trigger in the sentence. The first feature has two values, namely "before" (the participant appears before the trigger) or "after" (the participant appears after the trigger). The second feature encodes the distance between the trigger and the participant. Distance is measured as the number of tokens between the trigger and the participant. Our intuition is that, if a candidate trigger and participant are far away from each other, it is less likely that they characterize an event.

2.2.3 Dependency Relation Features

A dependency parse tree captures the semantic predicate-argument dependencies among the words of a sentence. Dependency tree paths between protein pairs have successfully been used to identify protein interactions (Bunescu and Mooney, 2007; Erkan et al., 2007). In this paper, we use the dependency paths to extract events. For a given trigger/participant pair, we extract the shortest path from the trigger to the participant, from the dependency parse of the sentence. We use the *McClosky-Charniak* parses which are converted to the Stanford Typed Dependencies format and provided to the participants by the shared task organizers. Previous approaches use both the words and the dependency relation types to represent the paths (Bunescu and Mooney, 2007; Erkan et al., 2007). Consider the dependency tree in Figure 1. The path from "phosphorylation" to "CD40" is "nsubj inhibits acomp binding prep_to domain num". Due to the large number of possible words, using the words on the paths might lead to data sparsity problems and to poor generalization. Suppose we have a sentence with similar semantics, where the synonym word "prevents" is used instead of "inhibits". If we use the words on the path to represent the path feature, we end up with two different paths for the two sentences that have similar semantics. Therefore, in this study we use only the dependency relation types among the words to represent the paths. For example, the path feature extracted for the (phosphorylation, CD40) negative trigger/participant pair is "nsubj acomp prep_to num" and the path feature extracted for the (phosphorylation, TRAF2) positive trigger/participant pair is "prep_of".

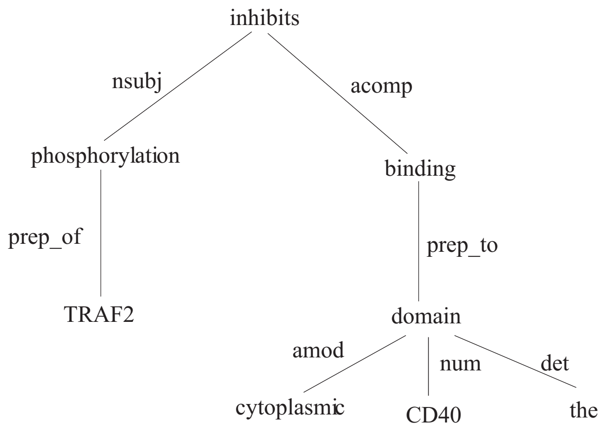


Figure 1: The dependency tree of the sentence "*The phosphorylation of TRAF2 inhibits binding to the CD40 cytoplasmic domain.*"

2.3 Classification

We used the SVM^{light} library (Joachims, 1999) with two different kernel functions and feature sets for learning the classification models. Our first approach is based on using linear SVM with the features described in Section 2.2. In this approach the path feature is used as a nominal feature. Our second approach is based on integrating to SVM a kernel function based on the word-based edit distance between the dependency relation paths, where each dependency relation type on the path is treated as a word. For example, the word-based edit distance between the paths "prep_of" and "prep_of prep_with" is 1, since 1 insertion operation (i.e., inserting "prep_with" to the first path) is sufficient to transform the first path to the second one. The edit-distance based similarity between two paths p_i and p_j and the corresponding kernel function are defined as follows (Erkan et al., 2007).

$$edit\_sim(p_i, p_j) = e^{-\gamma(edit\_distance(p_i, p_j))} \quad (1)$$

3 Experimental Results

The data provided for the shared task is prepared from the GENIA corpus (Kim et al., 2008). We used the training and the development sets for training.

The candidate triggers are detected by using a dictionary based approach, where the dictionary is extracted from the training set. We filtered out the noisy trigger candidates such as "with", "+", ":", and "-", which are rarely used as real triggers and commonly used in other contexts. The candidate trigger/participant pairs are classified by using the classifiers learned for Class 1, Class 2, and/or Class 3 depending on whether the candidate trigger matched one of the triggers in these classes. The SVM score is used to disambiguate the event types, if a candidate trigger matches a trigger in more than one of the event classes. A trigger which is ambiguous among the event types in the same class is assigned to the event type for which it is most frequently used as a trigger.

The results that we submitted to the shared task were obtained by using the linear SVM approach with the set of features described in Section 2.2. After submitting the results, we noticed that we made an error in pre-processing the data set. While aligning the provided dependency parses with the sentence, we incorrectly assumed that all the sentences had dependency parses and ended up using the wrong dependency parses for most of the sentences. The overall performance scores for our official submission are 30.42% recall, 14.11% precision, and 19.28% F-measure. The results obtained after correcting the error are reported in Table 1. Correcting the error significantly improved the performance of the system. Table 2 shows the results obtained by using SVM with dependency path edit kernel. The two SVM models achieve similar performances. The performance for the regulation events is considerably lower, since errors in identifying the events are carried to identifying the event participants of a regulation event. The performances for the events which have multiple participants, i.e., binding and regulation events, are lower compared to the events with a single participant. The performance is higher when computed by decomposing the events (49.00 and 31.82 F-measure for binding and regulation events, respectively). This suggests that even when participants of events are identified correctly, there is significant amount of error in composing the events.

Event Type	Recall	Precision	F-measure
Localization	41.95	60.83	49.66
Binding	31.41	34.94	33.08
Gene_expression	61.36	69.00	64.96
Transcription	37.23	30.72	33.66
Protein_catabolism	64.29	64.29	64.29
Phosphorylation	68.15	80.70	73.90
Event Total	50.82	56.80	53.64
Regulation	15.12	19.82	17.15
Positive_regulation	24.21	33.33	28.05
Negative_regulation	21.64	32.93	26.11
Regulation Total	22.02	30.72	25.65
All Total	35.86	44.69	39.79

Table 1: Approximate span & recursive matching results using linear SVM with the set of features described in Section 2.2 (after correcting the error in pre-processing the data set).

4 Conclusion

We described a supervised approach to extract biomolecular events. We grouped the event types into three general classes based on the number and types of participants that they can involve and learned separate SVM models for each class. We used various types of linguistic features that represent the context of the candidate event trigger/participant pairs. We achieved an F-measure of 39.83% on the shared task test data. Error analysis suggests that improving the approach of event composition for types of events with multiple participants and improving the strategy for detecting and disambiguating triggers can enhance the performance of the system.

Event Type	Recall	Precision	F-measure
Localization	49.43	64.18	55.84
Binding	31.70	35.03	33.28
Gene_expression	66.34	69.72	67.99
Transcription	39.42	25.59	31.03
Protein_catabolism	78.57	73.33	75.86
Phosphorylation	76.30	80.47	78.33
Event Total	55.13	56.62	55.86
Regulation	17.87	16.46	17.13
Positive_regulation	26.45	26.03	26.24
Negative_regulation	25.33	32.54	28.49
Regulation Total	24.68	25.34	25.01
All Total	39.31	40.37	39.83

Table 2: Approximate span & recursive matching results using SVM with dependency relation path edit kernel.

Acknowledgments

This work was supported in part by the NIH Grant U54 DA021519.

References

R. C. Bunescu and R. J. Mooney, 2007. *Text Mining and Natural Language Processing*, Chapter Extracting Relations from Text: From Word Sequences to Dependency Paths, pages 29–44, Springer.

Güneş Erkan, Arzucan Özgür, and Dragomir R. Radev. 2007. Semi-supervised classification for extracting protein interaction sentences using dependency parsing. In *Proceedings of EMNLP*, pages 228–237.

T. Joachims, 1999. *Advances in Kernel Methods-Support Vector Learning*, Chapter Making Large-Scale SVM Learning Practical. MIT-Press.

Jin-Dong Kim, Tomoko Ohta, and Jun'ichi Tsujii. 2008. Corpus annotation for mining biomedical events from literature. *BMC Bioinformatics*, 9(1).

Jin-Dong Kim, Tomoko Ohta, Sampo Pyysalo, Yoshinobu Kano, and Jun'ichi Tsujii. 2009. Overview of BioNLP'09 Shared Task on Event Extraction. In *Proceedings of Natural Language Processing in Biomedicine (BioNLP) NAACL 2009 Workshop*. To appear.

Biomedical Event Detection using Rules, Conditional Random Fields and Parse Tree Distances

Farzaneh Sarafraz*, James Eales*, Reza Mohammadi♦, Jonathan Dickerson♣, David Robertson♣, Goran Nenadic*

*School of Computer Science, University of Manchester

♣Faulty of Life Sciences, University of Manchester

♦Dept. of Mathematics and Computer Science, Sharif University of Technology

sarafraf@cs.man.ac.uk, g.nenadic@manchester.ac.uk

Abstract

This paper reports on a system developed for the BioNLP'09 shared task on detection and characterisation of biomedical events. Event triggers and types were recognised using a conditional random field classifier and a set of rules, while event participants were identified using a rule-based system that relied on relative distances between candidate entities and the trigger in the associated parse tree. The results on previously unseen test data were encouraging: for non-regulatory events, the F-score was almost 50% (with precision above 60%), with the overall F-score of around 30% (49% precision). The performance on more complex regulatory events was poor (F-measure of 7%). Among the 24 teams submitting the test results, our results were ranked 12th for the overall F-score and 8th for the F-score of non-regulation events.

1 Introduction

The aim of the BioNLP'09 shared task 1 was to characterise molecular events being reported in a Medline abstract by identifying the textual trigger, event type and participating entities (Kim et al. 2009). Nine event types were considered: *gene expression*, *transcription*, *protein catabolism*, *localisation*, *phosporylation*, *binding*, *regulation*, *positive regulation*, and *negative regulation*. Depending on the event type, the task included the identification of either one (for the first five event types mentioned above) or more (e.g. for *binding*) participating proteins. Information requested for regulatory events was more complex: in addition to one theme (a protein or another event), these events could also have a cause (a protein or another event) that needed to be identified.

The organisers have distributed a training dataset of 800 abstracts, with gene and gene product mentions pre-annotated in text. In addition, a development set (150 abstracts) was provided to assess the quality of the extractions during the training and development phases.

2 Methods

The system developed for the challenge consists of three main modules: (1) event trigger and type detection, (2) event participant detection, and (3) post-processing of the results.

2.1 Event Trigger and Type Detection

Our view of the event trigger and type detection subtask was that each token in a sentence needed to be tagged either as a trigger for one of the nine event types, or as a non-trigger/event token. We therefore decided to identify event types and triggers in a single step by training a conditional random field (CRF) classifier that assigned one of ten (nine types plus non-trigger) tags to each token. CRFs have been shown to be particularly suitable for tagging sequential data such as natural language text, because they take into account features and tags of neighbouring tokens when evaluating the probability of a tag for a given token.

Tokens and their part-of-speech (POS) tags were recognised using the Genia Tagger (Tsuruoka et al. 2005). Each stemmed token was represented using a feature vector consisting of the following features:

- A binary feature indicating whether the token is a protein;
- A binary feature indicating whether the token is a known protein-protein interaction word (we used a pre-complied dictionary of

such words collected from previous studies (Fu et al. 2008; Yang et al. 2008);

- The token's POS tag;
- The log-frequencies of the token being a trigger for each event type in the training data (nine features);
- The number of proteins in the given sentence.

Other features (e.g. separating the known interaction words according to the nine event types) were explored during the development phase, but were not included in the final feature list since they increased the sparseness of the data and did not improve the overall results. The CRF parameters were adjusted for maximum performance, including the choice of training algorithms, the number of training steps, the size of the window within which the tokens can affect any certain token, and the number of training abstracts used in each training step. It was interesting to notice that there were no significant improvements in the performance after training on 100, 400 or 800 abstracts from the training set (data not shown).

2.2 Locating Event Themes

After detecting potential triggers and associated event types, the next task was to locate possible participants (i.e. 'themes' and 'causes') for each event. It was obvious that participants did not have to be the nearest to the trigger on the surface level, so our approach was based on distances within the parse trees associated with the sentences containing candidate events. Parse tree distances have been studied previously in clustering and automatic translation tasks (Emms 2008), so we hypothesised that we could use them to identify the most likely participants. The training data was analysed for the proximities between the triggers and the (correct) event participants in the parse tree of the sentence.[1] Figure 1 gives a detailed density function of these distances (ignoring non-protein nodes). The analysis showed that a theme was usually amongst the nearest proteins to the trigger in terms of parse tree distances: for example, in 60% of all single theme events (e.g. *localisation*, *phosphorylation*) the correct protein participant was the trigger's nearest or second nearest protein in the parse tree. A further analysis demonstrated that it was more likely for a theme to appear in the sub-tree of the corresponding trigger, with 70% of all single theme events having a theme which appeared in the sub-tree of the trigger. Furthermore, specific analyses of the parse trees associated to the *binding* events (which can have more than one theme) suggested a linear relationship between the parse tree distance and binding event participant number (participant$_1$ is the nearest, participant$_2$ is the second nearest, etc.).

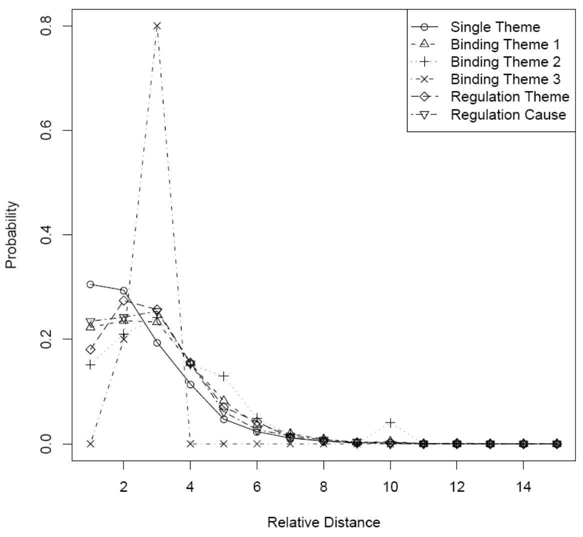


Figure 1: Probability density function of the distance between the trigger and the theme in the parse tree (ignoring the tokens that are not proteins)

We used this distributional analysis (derived from the training data) to design a rule-based method for the identification of participating themes. The rules were manually derived for each of the nine event classes, by defining:

- a threshold for the maximum distance to the trigger in the sub-tree for the given event type;
- a threshold for the difference between the maximum distance in the whole tree and the given sub-tree for the given event type;
- the number of nearest proteins to be reported for each trigger.

All entities that satisfied a distance-based rule for a given trigger were selected as the corresponding theme(s). For example, if the event type is *binding*, then up to the second closest protein in the sub-tree, and the first closest protein in the rest of the tree are reported as themes.

[1] The parse trees were produced by the GDep parser (Sagae and Tsujii, 2007) and supplied by the challenge organisers.

Figure 2 provides an example of the method applied to a sentence with multiple events. *Regulates* and *secretion* are correctly identified as triggers for a regulation and a localization event in the first phase. Using the rules for localization, the themes for two localization events are correctly recognised as proteins *T2* and *T3*, whereas *T1* was ignored since it did not appear in the trigger's sub-tree.

Engineering and applying rules for non-regulatory events was relatively straightforward. However, regulatory events can have different kinds of participants (a protein or an event). In the case of an event, we were trying to locate the nearest trigger for the event (being regulated) in the parse tree. For example, in Figure 2, the nearest option to the regulation trigger (*secretion*) was the trigger of the two localization events, and both events should be (correctly) reported as the themes of two regulation events. Therefore, we require a number of recursions in the application of the rules to represent higher-order regulatory dependences. For the purposes of this challenge, only regulations up to the second "order" were detected, allowing other events to act as themes and causes as well as proteins. Attempts to find more complicated regulatory events using this method resulted in a decreased precision and/or F-score.

2.3 Post-processing Event Profiles

The performance of the first two phases was studied on the development dataset: we noted a number of false-positive and false-negative results that were mostly due to a set of recurring triggers. We therefore decided to perform a post-processing step to improve the identification of event triggers and associated types. In the first step (improving the event trigger and type detection), the output of the CRF was overridden in cases where the triggers appeared in a list of negatively discriminated trigger words which was collected after the manual analysis of the false positive results on the training and development data. Similarly, in cases where the CRF missed a highly indicative trigger (from a manually collected set) for a given event type, the trigger was added as part of post-processing. In the latter case, the sentence was then processed for the event theme detection (as described in 2.2).

In the second step of the pre-processing phase, we forced highly indicative regulation triggers (if not previously identified) to be associated with an event by assigning proteins appearing in the sentence to them, even when no protein in the sentence satisfied the theme or cause criteria described in Section 2.2. This was aimed at improving the extremely low recall for regulatory events.

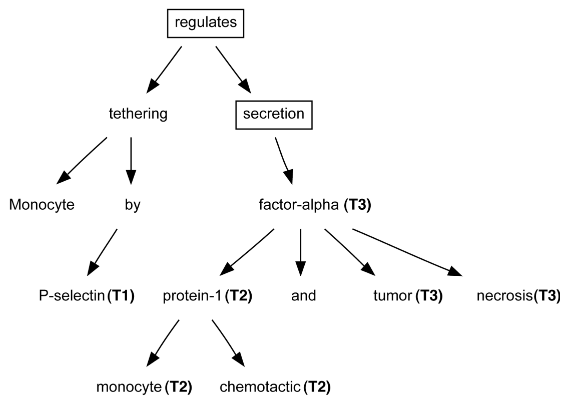


Figure 2: The parse tree of sentence *Monocyte tethering by P-selectin regulates monocyte chemotactic protein-1 and tumor necrosis factor-alpha secretion.* The triggers are shown in boxes, and the entities are numbered.

Finally, since triggers could consist of more than one consecutive token, a set of simple rules were applied to remove typical false-negative constituents identified by the CRF as part of triggers (e.g. sometimes linking words appeared within triggers).

3 Results and discussion

The task 1 assessment was based on the output of the system when applied to the test dataset of 260 previously unseen abstracts. An event was counted as a true positive if its type, trigger and all participants had been correctly identified. The overall F-score for our system was 30.35% with 48.61% precision (approximate span matching, see Table 1). The best performing event types were *phosphorylation* (the best F-score and the best recall) and *gene expression* (the best precision with a reasonably good F-measure). While the results for non-regulatory events were encouraging, they were low for regulatory events. Among the 24 teams submitting the test results, our results were ranked 12th for the overall F-score and 8th for the F-score of non-regulation events.

A preliminary analysis of the results was performed on the development data (as the test data is not available), which had around 5% higher overall F-score than the test data (9% for non-regulation events, see Table 2 for details).

Event Class	#Gold	R	P	F-score
Localisation	174	44.83	53.06	48.60
Binding	347	12.68	40.37	19.30
Gene expression	722	52.63	**69.34**	59.84
Transcription	137	15.33	67.74	25.00
Protein catabolism	14	42.86	50.00	46.15
Phosphorylation	135	**78.52**	53.81	**63.86**
Non-reg total	1529	41.53	60.82	49.36
Regulation	291	3.09	19.15	5.33
Positive regulation	983	1.12	8.87	1.99
Neg. regulation	379	12.4	20.52	15.46
Regulatory total	1653	4.05	16.75	6.53
All total	3182	22.06	48.61	30.35

Table 1: Evaluation of the test data (260 abstracts), (approximate span matching; #Gold = the number of examples in the gold standard)

Event Class	#Gold	R	P	F-score
Localisation	53	67.92	46.75	55.38
Binding	312	21.47	63.81	32.13
Gene expression	356	64.61	76.33	69.98
Transcription	82	53.66	**89.80**	67.18
Protein catabolism	21	90.48	67.86	**77.55**
Phosphorylation	47	**91.49**	53.09	67.19
Non-reg total	871	50.4	68.44	58.05
Regulation	172	5.23	33.33	9.05
Positive regulation	632	3.48	21.36	5.99
Neg. regulation	201	9.45	15.08	11.62
Regulatory total	1005	4.98	19.53	7.93
All total	1876	26.07	54.46	35.26

Table 2: Evaluation of the development data (150 abstracts) (approximate span matching; #Gold as in Table 1)

In order to assess the effects of different steps in our approach, we evaluated the performance of the event trigger and event participant detection steps separately. The results presented in Table 3 indicated that the performance of the CRF module was not much better than the overall performance of the system (an F-score of 43% vs. 35%), suggesting that the CRF part was mostly responsible for the errors, by both missing triggers and falsely reporting them. This was particularly the case with non-regulatory events (even for binding). Conversely, when considering only those events whose triggers were correctly identified, their participants were also correctly recognised in most cases. Overall, the analysis suggested that the parse tree distance method performed reasonable well, despite a reduction in recall of approximately 12%.

There are a number of possibilities for improvements. We believe applying the CRF model in two stages would be a better approach to detect events: first identify triggers and then link them to event classes. In addition, the rules employed for determining themes need to be more specific to reflect both event type and grammatical structure. In the case of regulatory events, however, significantly better results were noticed in the trigger detection part when compared to the overall scores, indicating that it was difficult to identify regulatory participants, as any of those participants could be either a protein or another event.

Event Class	#Gold	R	P	F-score
Localisation	40	77.50	47.69	59.05
Binding	180	33.33	54.55	41.38
Gene expression	282	76.60	58.54	66.36
Transcription	68	58.82	18.60	28.27
Protein catabolism	19	84.21	**88.89**	86.49
Phosphorylation	40	**97.50**	81.25	**88.64**
Non-reg total	629	63.91	48.73	55.30
Regulation	138	13.04	62.07	21.56
Positive regulation	462	13.85	54.24	22.07
Neg. regulation	153	29.41	45.92	35.86
All total	1382	38.28	49.44	43.15

Table 3: Trigger-only evaluation of the development data

Overall, the results achieved by our system suggest that combining parse tree results, rules and CRFs is a promising approach for the identification of non-regulatory events in the literature, while more work would be needed for regulatory events.

References

Emms M. 2008. *Tree-distance and some other Variants of evalb*. Proc. of LREC 2008, pp 1373-1379.

Fu W. *et al.* 2008. *Human Immunodeficiency Virus type 1, Human protein interaction database at NCBI*, Nucleic Acid Research 2008, D417-D422

Kim JD *et al.* 2009. *Overview of BioNLP'09 Shared Task on Event Extraction*, Proc. of BioNLP NAACL 2009 Workshop (to appear)

Sagae K, Tsujii J. 2007. *Dependency Parsing and Domain Adaptation with LR Models and Parser Ensembles*. Proc. of the CoNLL 2007 Shared Task, 1044-50

Tsuruoka Y. *et al.* 2005. *Developing a Robust Part of-Speech Tagger for Biomedical Text*. Advances in Informatics, 382–392.

Yang H. et al. 2008. *Identification of Transcription Factor Contexts in Literature using Machine Learning Approaches*. BMC Bioinformatics, Vol. 9(3):S11.

Syntactic Dependency Based Heuristics for Biological Event Extraction

Halil Kilicoglu and Sabine Bergler
Department of Computer Science and Software Engineering
Concordia University
1455 de Maisonneuve Blvd. West
Montréal, Canada
{h_kilico,bergler}@cse.concordia.ca

Abstract

We explore a rule-based methodology for the BioNLP'09 Shared Task on Event Extraction, using dependency parsing as the underlying principle for extracting and characterizing events. We approach the speculation and negation detection task with the same principle. Evaluation results demonstrate the utility of this syntax-based approach and point out some shortcomings that need to be addressed in future work.

1 Introduction

Exponential increase in the amount of genomic data necessitates sophisticated approaches to accessing knowledge in molecular biology literature, which remains the primary medium for disseminating new knowledge in molecular biology. Extracting relations and events directly from free text facilitates such access. Advances made in foundational areas, such as parsing and named entity recognition, boosts the interest in biological event extraction (Zweigenbaum et al., 2007). The BioNLP'09 Shared Task on Event Extraction illustrates this shift and is likely to inform future endeavors in the field.

The difficulty of extracting biological events from scientific literature is due to several factors. First, sentences are long and often have long-range dependencies. In addition, the biological processes described are generally complex, involving multiple genes or proteins as well as other biological processes. Furthermore, biological text is rich in higher level phenomena, such as speculation and negation, which need to be dealt with for correct interpretation of the text. Despite all this complexity, however, a closer look at various biological corpora also suggests that beneath the complexity lie regularities, which may potentially be exploited using relatively simple heuristics.

We participated in Task 1 and Task 3 of the Shared Task on Event Extraction. Our approach draws primarily from dependency parse representation (Mel'čuk, 1988; deMarneffe et al., 2006). This representation, with its ability to reveal long-range dependencies, is suitable for building event extraction systems. Dependencies typed with grammatical relations, in particular, benefit such applications. To detect and characterize biological events (Task 1), we constructed a dictionary of event triggers based on training corpus annotations. Syntactic dependency paths between event triggers and event participants in the training corpus served in developing a grammar for participant identification. For speculation and negation recognition (Task 3), we extended and refined our prior work in speculative language identification, which involved dependency relations as well. Our results show that dependency relations, despite their imperfections, provide a good foundation, on which accurate and reliable event extraction systems can be built and that the regularities of biological text can be adequately exploited with a limited set of syntactic patterns.

2 Related Work

Co-occurrence based approaches (Jenssen et al., 2001; Ding et al., 2002) to biological relation extraction provide high recall at the expense of low precision. Shallow parsing and syntactic templates (Blaschke et al., 1999; Rindflesch et al., 2000; Friedman et al., 2001; Blaschke and Valencia, 2001; Leroy et al., 2003; Ahlers et al., 2007), as well as full parsing (Daraselia et al., 2004; Yakushiji et al., 2005), have also been explored as the basis for relation extraction. In contrast to co-occurrence based methods, these more sophisticated approaches provide higher precision at the expense of lower recall. Approaches combining the strengths of complementary models have also been proposed (Bunescu et al., 2006) for high recall and precision.

More recently, dependency parse representation has found considerable use in relation extraction, particularly in extraction of protein-protein interactions (PPI). Fundel et al. (2007) use Stanford dependency parses of Medline abstracts as the basis for rules that extract gene/protein interactions. Rinaldi et al. (2007) extract relations combining a hand-written grammar based on dependency parsing with a statistical language model. Airola et al. (2008) extract protein-protein interactions from scientific literature using supervised machine learning based on an all-dependency-paths kernel.

The speculative aspect of the biomedical literature (also referred to as *hedging*) has been the focus of several recent studies. These studies primarily dealt with distinguishing speculative sentences from non-speculative ones. Supervised machine learning techniques mostly dominate this area of research (Light et al., 2004; Medlock and Briscoe, 2007; Szarvas, 2008). A more linguistically-based approach, relying on lexical and syntactic patterns, has been explored as well (Kilicoglu and Bergler, 2008). The scope of speculative statements is annotated in the BioScope corpus (Vincze et al., 2008); however, experiments in detecting speculation scope have yet to be reported.

Recognizing whether extracted events are negated is crucial, as negation reverses the meaning of a proposition. Most of the work on negation in the biomedical domain focused on finding negated terms or concepts. Some of these systems are rule-based and rely on lexical or syntactic information (Mutalik et al., 2001; Chapman et al., 2001; Sanchez-Graillet and Poesio, 2007); while others (Averbuch et al., 2004; Goldin and Chapman, 2003) experiment with machine learning techniques. A recent study (Morante et al., 2008) focuses on learning negation scope using memory-based classifiers trained on the BioScope corpus.

Our approach to Task 1 is most similar to work of Fundel et al. (2007) as it builds on dependency-based heuristics. However, we address a larger number of event classes, including regulatory events allowing participation of other events. In addition, event triggers are central to our approach, contrasting with their system and most other PPI systems that rely on finding dependency paths between entities. We extended prior work for Task 3 and obtained state of the art results.

3 Event Detection and Characterization

As preparation for biological event extraction, we combined the provided annotations, tokenized input and dependency parses in an XML representation. Next, we determined good trigger words for event classes and scored them. Finally, we developed a dependency-based grammar for event participant identification, which drives our event extraction system.

3.1 Data Preprocessing

Our event detection and characterization pipeline requires XML representation of a document as input. Here, the XML representation of a document contains sentences, their offset positions and dependency parses as well as entities (Proteins) and their offset positions in addition to word information (tokens, part-of-speech tags, indexes and lemmas). We used the Stanford Lexicalized Parser (Klein and Manning, 2003) to extract word-related information, as well as for dependency parsing.

3.2 Event Triggers

After parsing the training corpus and creating an enriched document representation, we proceeded with constructing a dictionary of event triggers, drawing from training corpus annotations of triggers and making further refinements, as described below.

We view event triggers essentially as predicates and thus restricted event triggers to words carrying verb, noun or adjective part-of-speech tags. Our analysis suggests that, in general, trigger words with other POS tags are tenuously annotated event triggers and in fact require more context to qualify as event triggers. In Example (1), *by* is annotated as trigger for a `Positive_regulation` event; however, it seems that the meaning of the entire prepositional phrase introduced with *by* contributes to trigger such an event:

(1) These data suggest a general role for Tax induction of IL-1alpha gene transcription *by* the NF-kappaB pathway.

We refined the event trigger list further through limited term expansion and filtering, based on several observations:

1. The event triggers with prefixes, such as *co*, *down* and *up*, (e.g., *coexpression*, *down-regulate*) were expanded to include both hyphenated and non-hyphenated forms.

2. For a trigger that has inflectional/derivational forms acting as triggers in the development corpus but not in the training corpus, we added these forms as event triggers. Examples include adding *dimerization* after *dimerize* and *diminished*(adj) after *diminish*, among others.

3. We removed several event triggers, which, we considered, required more context to qualify as event triggers for the corresponding event classes. (e.g., *absence*, *absent*, *follow*, *lack*)

Finally, we did not consider multi-word event triggers. We observed that core trigger meaning generally came from a single word token (generally head of a noun phrase) in the fragment annotated as event trigger. For instance, for trigger *transcriptional activation*, the annotated event class is `Positive_regulation`, which suggests that the head *activation* carries the meaning in this instance (since *transcriptional* is an event trigger for the distinct `Transcription` event class). In another instance, the trigger *binding activity* is annotated as triggering a `Binding` event, indicating that the head word *activity* is semantically empty. We noted some exceptions to this constraint (e.g., *negatively regulate*, *positive regulation*) and dealt with them in the postprocessing step.

For the remaining event triggers, we computed a "goodness score" via maximum likelihood estimation. For a given event class *C* and event trigger *t*, the "goodness score" *G(t,C)* then is:

$$G(t,C) = w(C{:}t)/w(t)$$

where *w(C:t)* is the number of times *t* occurs as a trigger for event class *C* and *w(t)* is the frequency of trigger *t* in the training corpus. The newly added event triggers were assigned the same scores as the trigger they are derived from.

In the event extraction step, we do not consider event triggers with a score below an empirically determined threshold.

3.3 Dependency relations for event participant identification

To identify the event participants Theme and Cause, we developed a grammar based on the "collapsed" version of Stanford Parser dependency parses of sentences. Grammar development was driven by extraction and ranking of typed dependency relation paths connecting event triggers to corresponding event participants in the training data. We then analyzed these paths and implemented as rules those deemed to be both correct and sufficiently general.

More than 2,000 dependency paths were extracted; however, their distribution was Zipfian, with approximately 70% of them occurring only once. We concentrated on the most frequent, therefore general, dependency paths. Unsurprisingly, the most frequent dependency path involved the *dobj* (direct object) dependency between verbal event triggers and Theme participants, occurring 826 times. Next was the *nn* (nominal modifier) dependency between nominal event triggers and their Theme participants. The most frequent dependency for Cause participants was, again unsurprisingly, *nsubj* (nominal subject). The ranking of dependency paths indicated that path length is inversely proportional to reliability. We implemented a total of 27 dependency path patterns.

Some of these patterns specifically address deficiencies of the Stanford Parser. Prepositional phrases are often attached incorrectly, causing problems in participant identification. Consider, for example, one of the more frequent dependency paths, *dobj-prep_on* (direct object dependency followed by prepositional modifier headed in *on*), occurring between the event trigger (*effect*) and participant (*expression*, itself a sub-event trigger):

(2) We have examined the *effect* of leukotriene B4

(LTB4), a potent lipid proinflammatory mediator, *on* the *expression* of the proto-oncogenes c-jun and c-fos.
dobj(examined,effect)
prep_on(examined,expression)

This dependency path occurs almost exclusively with PP attachment errors involving *on*, leading us to stipulate a "corrective" dependency path, implemented for certain trigger words (e.g., *effect*, *influence*, *impact* in this case). Postnominal prepositional attachment heuristics detailed in Schuman and Bergler (2006) helped determine 6 such patterns.

Two common verbs (*require* and *involve*) deserve special attention, as the semantic roles of their subject/object constituents differ from typical verbs. The prototypical Cause dependency, *nsubj*, indicates a Theme in the following sentence:

(3) *Regulation* of interleukin-1beta transcription by Epstein-Barr virus *involves* a number of latent proteins via their interaction with RBP.
nsubj(involves,Regulation)

For these two verbs, participant identification rules are reversed.

An interesting phenomenon is NPs with hyphenated adjectival modifiers, occurring frequently in molecular biology texts (e.g., "... *LPS-mediated TF expression*..."). The majority of these cases involve regulatory events. Such cases do not involve a dependency path, as the participant (in this case, *LPS*) and the event trigger (*mediated*) form a single word. An additional rule addresses these cases, stipulating that the substring preceding the hyphen is the Cause of the regulatory event triggered by the substring following the hyphen. (`Positive_regulation` (Trigger=*mediated*,Theme=*TF expression*,Cause=*LPS*)).

Events allowing event participants (regulatory events) are treated essentially the same way as events taking entity participants. The main difference is that, when sub-events are considered, a dependency path is found between the trigger of the main event and the trigger of its sub-event, rather than an annotated entity, as was shown above in Example (2).

3.4 Extracting Events

The event detection and characterization pipeline (Task 1) consists of three steps:

1. Determining whether a word is an event trigger.
2. If the word is an event trigger, identifying its potential participant(s).
3. If the event trigger corresponds to a regulatory event and it has a potential sub-event participant, determining in a recursive fashion whether the sub-event is a valid event.

The first step is a simple dictionary lookup. Provided that a word is tagged as noun, verb or adjective, we check whether it is in our dictionary, and if so, determine the event class for which it has a score above the given threshold. This word is considered the clue for an event.

We then apply our dependency-based rules to determine whether any entity or event trigger (in the case of regulatory events) in the sentence qualifies as an argument of the event clue. Grammar rules are applied in the order of simplicity; rules that involve a direct dependency between the clue and any word of the entity are considered first.

Once a list of potential participants is obtained by consecutive application of the rules, one of two things may happen: Provided that sub-events are not involved and appropriate participants have been identified (e.g., a Theme is found for a `Localization` event), the event is simply added to the extracted event list. Otherwise, we proceed recursively to determine whether the sub-event participant can be resolved to a simple event. If this yields no such simple event in the end, the event in question is rejected. In the following example, the event triggered by *inhibit* is invalid even though its Cause *JunB* is recognized, because its Theme, sub-event triggered by *activation*, cannot be assigned a Theme and therefore is considered invalid.

(4) ..., *JunB*, is shown to *inhibit activation* mediated by JunD.

After events are extracted in this manner, two postprocessing rules ensure increased accuracy. One rule deals with a limited set of multi-word event triggers. If a `Regulation` event has been identified and the event trigger is modified by *positive* or *negative* (or inflectional forms *positively*, *negatively*), the event class is updated to `Positive_regulation` or `Negative_regulation`, respectively. The second rule deals with the limitation of not allowing multiple events on the same trigger and adds to the extracted event list a `Positive_regulation` event, if a `Gene_expression` event was recognized for certain triggers, including *overexpression* and several others related to *transfection* (e.g., *transfect, transfection, cotransfect*).

Two grammatical constructions are crucial to determining the event participants: coordination and apposition. We summarize how they affect event extraction below.

3.4.1 Coordination

Coordination plays two important roles in event extraction:

1. When the event trigger is conjoined with another word token, dependency relations concerning the other conjunct are also considered for participant identification.
2. When an event is detected and its participant is found to be coordinated with other entities, new events are created with the event trigger and each of these entities. An exception are `Binding` events, which may have multiple Themes. In this case, we add conjunct entities as the Themes of the base event.

Coordination between words is largely determined by dependency relations. The participants of a dependency with a type descending from *conj* (conjunct) are considered coordinated (e.g., *conj_and*, *conj_or*).

Recognizing that Stanford dependency parsing misses some expressions of coordinated entities typical of biological text (in particular, those involving parentheses), we implemented a few additional rules to better resolve coordinated entities. These rules stipulate that entities that have between them:

1. Only a comma (,) or a semi-colon (;)
2. A word with CC (coordinating conjunction) part-of-speech tag
3. A complete parenthetical expression
4. Any combination of the above

are coordinated. For instance, in Example (5), we recognize the coordination between *interleukin-2* and *IL-4*, even though the parser does not:

(5) The activation of NFAT by TCR signals has been well described for *interleukin-2* (IL-2) and *IL-4* gene transcription in T cells.
conj_and(interleukin-2,transcription)

3.4.2 Apposition

Words in an apposition construction are considered equivalent for event extraction purposes. Therefore, if an appropriate dependency exists between a word and the trigger and the word is in apposition with an entity, that entity is marked as the event participant. In Example 6, the *appos* (appositive) dependency shown serves to extract the event `Positive_regulation` (Trigger=*upregulation*, Theme=*intercellular adhesion molecule-1*):

(6) ... *upregulation* of the lung vascular adhesion molecule, *intercellular adhesion molecule-1*, was greatly reduced by...
appos(molecule,molecule-1)
prep_of(upregulation,molecule)

The dependencies that we consider to encode apposition constructions are: *appos* (appositive), *abbrev* (abbreviation), *prep_{including, such_as, compared_to, compared_with, versus}* (prepositional modifier marked with *including*, *such as*, *compared to*, *compared with* or *versus*).

3.5 Speculation and Negation Detection

Once an event list is obtained for a sentence, our speculation and negation module determines whether these events are speculated and/or negated, using additional dependency-based heuristics that consider the dependencies between the event trigger and speculation/negation cues.

3.5.1 Speculation Recognition

We refined an existing speculation detection module in two ways for Task 3. First, we noted that modal verbs (e.g., *may*) and epistemic adverbs (e.g., *probably*) rarely mark speculative contexts in the training corpus, demonstrating the lack of a standardized notion of speculation among various corpora. For Task 3, we ignored lexical cues in these classes completely for increased accuracy. Secondly, corpus analysis revealed a new syntactic pattern for speculation recognition. This pattern involves the class of verbs that we called *active cognition verbs* (e.g., *examine*, *evaluate*, *analyze*, *study*, *investigate*). We search for a Theme dependency pattern between one of these verbs and an event trigger and mark the event as speculated, if such a pattern exists. Nominalizations of these verbs are also considered. In Example (7), the event triggered by *effects* is speculated, since *effects* is the direct object (therefore, Theme) of *studied*:

(7) We have *studied* the *effects* of prednisone (PDN), ... on the production of cytokines (IL-2, IL-6, TNF-alpha, IL-10) by peripheral T lymphocytes...

3.5.2 Negation Detection

Negation detection is similar to speculation detection. Several classes of negation cues have been determined based on corpus analysis and the negation module negates events if there is an appropriate dependency between one of these cues and the event triggers. The lexical cues and the dependencies that are sought are given in Table 1.

Negation Cue	**Dependency**
lack, absence	*prep_of(Cue,Trigger)*
unable, <not> able, fail	*xcomp(Cue,Trigger)*
inability, failure	*infmod(Cue, Trigger)*
no, not, cannot	*det(Trigger, Cue)*

Table 1: Negation cues and the corresponding dependencies (*xcomp*: clausal complement, *infmod*: infinitival modifier, *det*: determiner)

Additionally, participation of event triggers in dependencies of certain types is sufficient for negating the event it triggers. Such dependency types are *neg* (negation) and *conj_negcc* (negated coordination). A *neg* dependency applies to event triggers only, while *conj_negcc* is sought between event participants, as well as event triggers. Therefore, in Example (8), an event (`Positive_regulation`(Trigger=*transactivate*, Theme: *GM-CSF*, Cause=*ELF1*)) is negated, based on the dependencies below:

(8) Exogenous ETS1, but not *ELF1*, can *transactivate GM-CSF*, ..., in a PMA/ionomycin dependent manner.
conj_negcc(ETS1, ELF1)
nsubj(transactivate, ETS1)
dobj(transactivate, GM-CSF)

Finally, if none of the above applies and the word preceding the event trigger or one of the event participants is a negation cue (*no, not, cannot*), the event is negated.

4 Results and Discussion

Our event extraction system had one of the best performances in the shared task. With the approximate span matching/approximate recursive matching evaluation criteria, in Task 1, we were ranked third, while our speculation and negation detection module performed best among the six participating systems in Task 3. Not surprisingly, our system favors precision, typical of rule-based systems. Full results are given in Table 2.

The results reported are at goodness score threshold of .08. Increasing the threshold increases precision, while lowering recall. The threshold was determined empirically.

Our results confirm the usefulness of dependency relations as foundation for event extraction systems. There is much room for improvement, particularly in terms of recall, and we believe that incremental nature of our system development accommodates such improvements fairly easily.

Our view of event triggers ("once a trigger, always a trigger"), while simplistic, provides a good starting point by greatly reducing the number of trigger candidates in a sentence and typed dependencies to consider. However, it also leads to errors. One such example is given in Example (9):

(9) We show that ..., and that LPS treatment enhances the *oligomerization* of *TLR2*.

where we identify the event `Binding` (Trigger=*oligomerization*,Theme=*TLR2*). We consider *oligomerization* a reliable trigger, since it occurs twice in the training corpus, both times as event triggers. However, in this instance, it does not trigger

Event Class	Recall	Precis.	F-score
Localization	35.63	92.54	51.45
Binding	20.46	40.57	27.20
Gene_expression	55.68	79.45	65.47
Transcription	15.33	60.00	24.42
Protein_catabolism	64.29	56.25	60.00
Phosphorylation	69.63	95.92	80.69
EVT-TOTAL	43.10	73.47	54.33
Regulation	24.05	45.75	31.53
Positive_regulation	28.79	50.45	36.66
Negative_regulation	26.65	51.53	35.13
REG-TOTAL	27.47	49.89	35.43
Negation	14.98	50.75	23.13
Speculation	16.83	50.72	25.27
MOD-TOTAL	15.86	50.74	24.17
ALL-TOTAL	32.68	60.83	42.52

Table 2: Evaluation results

an event. This narrow view also leads to recall errors, in which we do not recognize an event trigger as such, simply because we have not encountered it in the training corpus, or it does not have an appropriate part-of-speech tag. A more sophisticated trigger learning approach could aid in better detecting event triggers.

We dealt with some deficiencies of Stanford dependency parsing through additional rules, as described in Section 3.3. However, many dependency errors are still generated, due to the complexity of biological text. For instance, in Example (10), there is a coordination construction between *NF-kappaB nuclear translocation* and *transcription of E-selectin and IL-8*. However, this construction is missed and an erroneous *prep_of* dependency is found, leading to two false positive errors: `Localization` (Trigger=*translocation*, Theme=*E-selectin*) and `Localization` (Trigger=*translocation*, Theme=*IL-8*).

(10) ... leading to NF-kappaB nuclear *translocation* and *transcription* of *E-selectin* and *IL-8*, which results in ...
conj_and(transcription, translocation)
prep_of(translocation, E-selectin)
conj_and(E-selectin, IL-8)

These errors can be corrected via other "corrective" dependency paths; however, first, a closer examination of such error patterns is necessary.

In other instances, the required dependency is completely missed by the parser, leading to recall errors. For instance, in Example (11), we are unable to recognize two events (`Regulation` (Trigger=*regulation*, Theme=*4E-BP1*) and `Regulation` (Trigger=*regulation*, Theme=*4E-BP2*)), due to lack of apposition dependencies between *repressors* and *4E-BP1* or *4E-BP2*:

(11) ... specific *regulation* of two repressors of translation initiation, *4E-BP1* and *4E-BP2*.
prep_of(regulation,repressors)
prep_of(repressors, initiation)
conj_and(intiation, 4E-BP1)
conj_and(initiation, 4E-BP2)

Typical of rule-base systems, we miss events expressed using less frequent patterns. Event participants expressed as prepositional modifiers marked with *from* is one such case. An example is given below:

(12) Calcineurin activates *transcription* from the *GM-CSF* promoter ...

In this case, the event `Transcription` (Trigger=*transcription*, Theme=*GM-CSF*) is missed. It is fairly easy to add a rule to address such occurrences.

We have not attempted to resolve anaphoric expressions for the shared task, which led to a fair number of recall errors. In a similar vein, we ignored events spanning multiple sentences. We expect that several studies addressing anaphora resolution in biomedical text (Castaño et al., 2002; Gasperin and Briscoe, 2008) will inform our near future efforts in this area.

Evaluation results regarding Task 3 may seem poor at first; however, most of the errors concern misidentified or missed base events. Thus, in this section, we focus on errors specifically triggered by speculation and negation module. In the development corpus, we identified 39 speculation instances, 4 of which were errors due to speculation processing. Of 95 annotated speculation instances, 7 were missed due to deficiencies in speculation processing.

Similarly, negation processing led to 5 false positives in 31 negation instances we identified and to 5 false negatives in 107 annotated negation instances.

We found that speculation false positive errors are exclusively cases for which speculation could be argued. For instance, in Example (13), we recognize that *appears to* scopes over event `Negative_regulation` (Trigger=*negatively regulate*, Theme=*IL-2R*), rendering it speculative. However, it is not annotated as such. This is further evidence for the difficulty of annotating such phenomena correctly and consistently, since the exact meaning is somewhat elusive.

(13) An unidentified Ets family protein binds to the EBS overlapping the consensus GAS motif and *appears to negatively regulate* the human IL-2R alpha promoter.

Negation pattern that involves negation cues (*no,not,cannot*) in the token preceding an event trigger or participant, a pattern initially considered to increase recall, caused most of negation false positive errors. An example is given in (14):

(14) The finding that HL-60/vinc/R cells respond to TPA with induction of a monocytic phenotype, but *not c-jun expression*, suggests that ...

Complex and less frequent patterns of expressing speculation and negation were responsible for more recall errors. Two such examples are given below:

(15) (a) These results ... and *suggest* a molecular mechanism for the *inhibition* of TLR2 by DN variants.

(b) *Galectin-3* is ... and is *expressed* in many leukocytes, *with the notable exception* of B and T lymphocytes.

In (15a), speculation is detected; however, we are unable to recognize that it scopes over the event triggered by *inhibition*. In (15b), the prepositional phrase, *with the notable exception*, is not considered to indicate negation.

5 Conclusions and Future Work

We explored a rule-based approach to biological event detection driven by typed dependency relations. This study marks our first foray into bio-event extraction in a general way and, thus, we consider the results very encouraging. In one area we investigated before, speculation detection, our system performed best and this confirms the portability and extensibility of our approach.

Modest recall figures point to areas of improvement. We plan to address anaphora resolution and multiple sentence spanning events in the near future. Our naïve approach to event triggers needs refinement and we believe that sophisticated supervised machine learning techniques may be helpful. In addition, biomedical lexical resources, including UMLS SPECIALIST Lexicon (McCray et al., 1994), may be useful in improving event trigger detection. Finally, dependency relations based on the Stanford Parser provided better performance in our case, in contrast to general consensus that those based on Charniak Parser (Charniak and Johnson, 2005) are superior, and this, too, deserves further investigation.

References

C B Ahlers, M Fiszman, D Demner-Fushman, F M Lang, and T C Rindflesch. 2007. Extracting semantic predications from Medline citations for pharmacogenomics. *Pac Symp Biocomput*, pages 209–220.

A Airola, S Pyysalo, J Björne, T Pahikkala, F Ginter, and T Salakoski. 2008. All-paths graph kernel for protein-protein interaction extraction with evaluation of cross-corpus learning. *BMC Bioinformatics*, 9 Suppl 11:s2.

M Averbuch, T Karson, B Ben-Ami, O Maimon, and L Rokach. 2004. Context-sensitive medical information retrieval. In *Proc MEDINFO-2004*, pages 1–8.

C Blaschke and A Valencia. 2001. The potential use of SUISEKI as a protein interaction discovery tool. *Genome Inform*, 12:123–134.

C Blaschke, M A Andrade, C Ouzounis, and A Valencia. 1999. Automatic extraction of biological information from scientific text: protein-protein interactions. In *Proc Int Conf Intell Syst Mol Biol*, pages 60–67.

R Bunescu, R Mooney, A Ramani, and E Marcotte. 2006. Integrating co-occurrence statistics with information extraction for robust retrieval of protein interactions from Medline. In *Proc BioNLP Workshop on Linking Natural Language Processing and Biology*, pages 49–56.

J Castaño, J Zhang, and J Pustejovsky. 2002. Anaphora resolution in biomedical literature. In *Proc International Symposium on Reference Resolution for NLP*.

W W Chapman, W Bridewell, P Hanbury, G F Cooper, and B G Buchanan. 2001. A simple algorithm for identifying negated findings and diseases in discharge summaries. *J Biomed Inform*, 34(5):301–310.

E Charniak and M Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proc 43rd Meeting of the Association for Computational Linguistics*, pages 173–180.

N Daraselia, A. Yuryev, S Egorov, S Novichkova, A Nikitin, and I Mazo. 2004. Extracting human protein interactions from MEDLINE using a full-sentence parser. *Bioinformatics*, 20(5):604–611.

M C deMarneffe, B MacCartney, and C D Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proc 5th International Conference on Language Resources and Evaluation*, pages 449–454.

J Ding, D Berleant, D Nettleton, and E Wurtele. 2002. Mining MEDLINE: abstracts, sentences, or phrases? *Pac Symp Biocomput*, 7:326–337.

C Friedman, P Kra, M Krauthammer, H Yu, and A Rzhetsky. 2001. GENIES: a natural-langauge processing system for the extraction of molecular pathways from journal articles. *Bioinformatics*, 17(1):74–82.

K Fundel, R Küffner, and R Zimmer. 2007. RelEx relation extraction using dependency parse trees. *Bioinformatics*, 23(3):365–371.

C Gasperin and T Briscoe. 2008. Statistical anaphora resolution in biomedical texts. In *Proc COLING 2008*.

I M Goldin and W W Chapman. 2003. Learning to detect negation with not in medical texts. In *Proc Workshop on Text Analysis and Search for Bioinformatics at the 26th ACM SIGIR Conference*.

T K Jenssen, A Laegreid, J Komorowski, and E Hovig. 2001. A literature network of human genes for high-throughput analysis of gene expression. *Nat Genet*, 28:21–28.

H Kilicoglu and S Bergler. 2008. Recognizing speculative language in biomedical research articles: a linguistically motivated perspective. *BMC Bioinformatics*, 9 Suppl 11:s10.

D Klein and C D Manning. 2003. Accurate unlexicalized parsing. In *Proc 41th Meeting of the Association for Computational Linguistics*, pages 423–430.

G Leroy, H Chen, and J D Martinez. 2003. A shallow parser based on closed-class words to capture relations in biomedical text. *Journal of Biomedical Informatics*, 36:145–158.

M Light, X Y Qiu, and P Srinivasan. 2004. The language of bioscience: facts, speculations, and statements in between. In *BioLINK 2004: Linking Biological Literature, Ontologies and Databases*, pages 17–24.

A T McCray, S Srinivasan, and A C Browne. 1994. Lexical methods for managing variation in biomedical terminologies. In *Proc 18th Annual Symposium on Computer Applications in Medical Care*, pages 235–239.

B Medlock and T Briscoe. 2007. Weakly supervised learning for hedge classification in scientific literature. In *Proc 45th Meeting of the Association for Computational Linguistics*, pages 992–999.

I A Mel'čuk. 1988. *Dependency syntax: Theory and Practice*. State University Press of New York, NY.

R Morante, A Liekens, and W Daelemans. 2008. Learning the scope of negation in biomedical text. In *Proc Conference on Empirical Methods in Natural Language Processing*, pages 715–724.

P G Mutalik, A Deshpande, and P M Nadkarni. 2001. Use of general-purpose negation detection to augment concept indexing of medical documents: A quantitative study using the UMLS. *J Am Med Inform Assoc*, 8(6):598–609.

F Rinaldi, G Schneider, K Kaljurand, M Hess, C Andronis, O Konstandi, and A Persidis. 2007. Mining of relations between proteins over biomedical scientific literature using a deep-linguistic approach. *Artif. Intell. Med.*, 39(2):127–136.

T C Rindflesch, L Tanabe, J N Weinstein, and L Hunter. 2000. EDGAR: Extraction of drugs, genes, and relations from the biomedical literature. In *Proc Pacific Symposium on Biocomputing*, pages 514–525.

O Sanchez-Graillet and M Poesio. 2007. Negation of protein protein interactions: analysis and extraction. *Bioinformatics*, 23(13):424–432.

J Schuman and S Bergler. 2006. Postnominal prepositional phrase attachment in proteomics. In *Proc BioNLP Workshop on Linking Natural Language Processing and Biology*, pages 82–89.

G Szarvas. 2008. Hedge classification in biomedical texts with a weakly supervised selection of keywords. In *Proc 46th Meeting of the Association for Computational Linguistics*, pages 281–289.

V Vincze, G Szarvas, R Farkas, G Mora, and J Csirik. 2008. The BioScope corpus: biomedical texts annotated for uncertainty, negation and their scopes. *BMC Bioinformatics*, 9 Suppl 11:S9.

A Yakushiji, Y Miyao, Y Tateisi, and J Tsujii. 2005. Biomedical event extraction with predicate-argument structure patterns. In *Proc First International Symposium on Semantic Mining in Biomedicine*, pages 60–69.

P Zweigenbaum, D Demner-Fushman, H Yu, and K B Cohen. 2007. Frontiers of biological text mining: current progress. *Briefings in Bioinformatics*, 8(5):358–375.

Analyzing text in search of bio-molecular events: a high-precision machine learning framework

Sofie Van Landeghem[1,2], **Yvan Saeys**[1,2], **Bernard De Baets**[3], **Yves Van de Peer**[1,2]
1. Dept. of Plant Systems Biology, VIB
2. Dept. of Plant Biotechnology and Genetics, Ghent University
3. Dept. of Applied Mathematics, Biometrics and Process Control, Ghent University
B-9000 Gent, Belgium
`yves.vandepeer@psb.vib-ugent.be`

Abstract

The BioNLP'09 Shared Task on Event Extraction is a challenge which concerns the detection of bio-molecular events from text. In this paper, we present a detailed account of the challenges encountered during the construction of a machine learning framework for participation in this task. We have focused our work mainly around the filtering of false positives, creating a high-precision extraction method. We have tested techniques such as SVMs, feature selection and various filters for data pre- and post-processing, and report on the influence on performance for each of them. To detect negation and speculation in text, we describe a custom-made rule-based system which is simple in design, but effective in performance.

1 Introduction

BioNLP recently emerged from the combined expertise of molecular biology and computational linguistics. At first, the community was mainly focused on named entity recognition (NER) and simple relation extraction, such as protein-protein interactions (Plake et al., 2005; Giuliano et al., 2006; Fundel et al., 2007; Saetre et al., 2008). However, the future of BioNLP lies in the ability to extract more complex events from text, in order to fully capture all available information (Altman et al., 2008).

Two recent community-wide challenges, Biocreative I (Hirschman et al., 2005) and II (Krallinger et al., 2008) have shown their merits by providing common benchmarking data and a meaningful comparison of various techniques. In contrast to the monolithic Biocreative tasks, the BioNLP'09 Shared Task has a more modular nature (Kim et al., 2009). It is not concerned with named entity recognition or normalization, but focuses on the task of event extraction itself.

This article is organized as follows: we first describe the Shared Task in a little more detail. Next, we present the methods used in our machine learning framework, carefully discussing our choices in design and their influence on performance. We then present the final results of our approach. Finally, we draw conclusions from our participation in this task, and suggest some future work for our own research as well as on a community-wide level.

2 BioNLP'09 Shared Task

2.1 Subtasks

The BioNLP'09 Shared Task was divided into three subtasks, of which only the first one was mandatory. We have participated in tasks 1 and 3, and will therefore only briefly discuss task 2. In accordance with the provided gold entity annotation, we will refer to all genes and gene products as *proteins*.

Task 1 represents the core of the challenge: detection and characterization of bio-molecular events from text. There are 9 distinct event types. Six events influence proteins directly, and we will refer to them as 'Protein events'. Five of them are unary: Localization, Gene expression, Transcription, Protein catabolism and Phosphorylation. The Binding event can be related to one protein (e.g. protein-DNA binding), two proteins (e.g. protein-protein interaction) or more (e.g. a complex). On top of these event types, there are three Regulation events: Regulation, Positive regulation and Negative regulation. Each of them can be unary or binary. In the latter case, an extra argument specifying the cause of the regulation is added. Each argument of a Regulation event can be either a protein or any other event.

Participants in task 2 had to recognise extra arguments for the events from task 1. For example, the cellular location should be added to a Localization event, and Site arguments had to be specified for Phosphorylation, Binding and Regulation.

Finally, task 3 was about detecting negation and speculation in text.

2.2 Examples

Suppose we are dealing with this sentence:

> "MAD-3 masks the nuclear localization signal
> of p65 and inhibits p65 DNA binding."

There are three proteins in this sentence:

- T1 : Protein : 'MAD-3'
- T2 : Protein : 'p65' (first occurrence)
- T3 : Protein : 'p65' (second occurrence)

There are also three triggers, which are defined by a contiguous stream of characters from the original text, and point to a specific event type:

- T27 : Negative regulation : 'masks'
- T29 : Negative regulation : 'inhibits'
- T30 : Binding : 'binding'

In this example, we see there is one binding event which involves trigger T30 and protein T3. Furthermore, this binding event is being influenced by protein T1, using trigger T29 which implies a Negative regulation event. Similarly, T1 has a negative effect on protein T2, which is expressed by trigger T27. When participating in subtask 2, one should also find the extra Site argument T28 for this last event:

- T28 : Entity : 'nuclear localization signal'

Now look at the following example:

> "NF-kappa B p50 is not directly regulated by
> I kappa B."

This sentence expresses a Regulation event involving the trigger 'regulated' and protein 'p50'. Participation in subtask 3 requires detecting the negation of this event.

2.3 Datasets

Both training and testing data consist of PubMed abstracts extracted from the GENIA corpus (Kim et al., 2008). All proteins are annotated and extra information is provided, such as analysis of sentence segmentation and tokenization, dependency graphs and phrase structure parses.

The training data consists of 800 articles. The development data contains an additional 150 articles with gold standard annotations. During development (6 weeks), the system's performance could be estimated with this dataset, using an online submission system. Participants had one week time to provide predictions for the final test dataset of 260 articles.

3 Methods

Our machine learning framework is tailored towards specific properties of different events, but is still kept sufficiently general to deal with new event types. The nature of the event extraction task leads to unbalanced datasets, with much more negative examples than positive ones. This is due to the fact that proteins could be involved in all possible event types, and each of the words in the text could be a trigger for an event. Finding the right events thus seems like looking for a needle in a haystack, which is why it is crucial to start with a good definition of candidate instances. This problem has motivated us to try and filter out as many irrelevant negative instances as possible by introducing specific preprocessing methods and filters. This reduces unbalancedness of the datasets and will lead to better precision as there will be less false positives (FPs). High-precision systems produce less noise and can be considered to be more useful when a researcher is trying to extract reliable interaction networks from text. There is a considerable degree of information redundancy in the original PubMed articles, which makes up for low recall when using the system in a real-world application. We have also tested a few post-processing techniques in order to remove FPs after classification.

Figure 1 shows a high-level overview of the different modules in our framework. More details are described in the next sections.

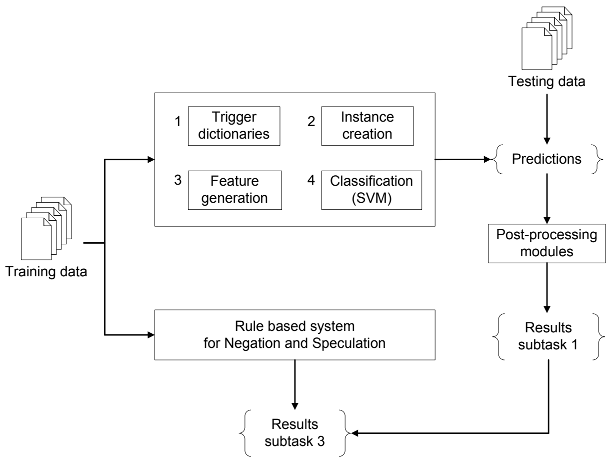


Figure 1: High-level overview of the modules used in our framework.

3.1 Parsing

For sentence segmentation, we made use of the provided tokenization files. Analysis of part-of-speech tags and dependency graphs was done using the Stanford parser (de Marneffe et al., 2006).

3.2 Dictionaries of triggers

From the training data, we automatically compiled dictionaries of triggers for each event type, applying the Porter stemming algorithm (Porter, 1980) to each trigger. This resulted in some entries in the dictionaries which were of limited use, such as 'through' for Binding, or 'are' for Localization. Such words are too general or too vague, and lead to many negative and irrelevant instances. For this reason, we *manually* cleaned the dictionaries, only keeping specific triggers for each event type (e.g. 'interaction' for Binding and 'secretion' for Localization).

During development, we noticed a significant difference between the triggers for unary Binding events (e.g. 'homodimer', 'binding site') and those for Binding events with multiple arguments (e.g. 'heterodimer', 'complex'). This motivated our choice to create two separate dictionaries and classifiers, thus discarding irrelevant candidate instances. Such an example would be a candidate binary Binding event with the trigger 'homodimer', while homodimerization is clearly a unary event. In the rest of this article, we will refer to these two event types as Single binding and Multiple binding events. The revision of the dictionaries resulted in a significant drop in the number of Binding instances in the training data, and improved the balancedness of the datasets: from a total of 34 612 instances (of which 2% positives) to 4708 Single binding instances (11% positives) and 3861 Multiple binding instances (5% positives).

Following the same reasoning, Regulation was also divided into unary and binary events. Furthermore, we have carefully analysed the nature of Binary regulation events, and noticed that a vast majority of these events had a protein in the 'cause' slot. We decided to split up the dictionaries of Binary regulations accordingly, differentiating between regulation events caused by proteins and those caused by other events. This keeps the more general words (e.g. 'causes') out of the dictionaries of events regulated by proteins (e.g. 'response'), again resulting in better balance of the datasets.

3.3 Instance creation

In a machine learning framework, a classifier tries to distinguish between positive instances (true biomolecular events) and negative instances (candidates which should be discarded). To run such a framework, one has to define candidate instances automatically by scanning the text. The first step towards instance creation consists of looking up triggers in text, using the constructed dictionaries for each event type. To this end, we have implemented a fast algorithm using Radix trees[1]. Next, candidate arguments have to be found. Initially, we have selected all (combinations of) proteins that were mentioned in the same sentence. However, this may result in a lot of negative and irrelevant instances, mainly in long sentences. This is why we have implemented a Negative-instances (NI) filter, which checks whether the length of the sub-sentence spanned by a candidate event does not exceed a certain value. Figure 2 shows the distribution of positive and negative Multiple binding events, according to the length of the relevant sub-sentence. It seems reasonable to only keep instances with a sub-sentence of less than 175 characters, as this includes almost all positive examples, while at the same time removing a significant amount of irrelevant negatives.

[1]Java implementation by Tahseen Ur Rehman, `http://code.google.com/p/radixtree/`

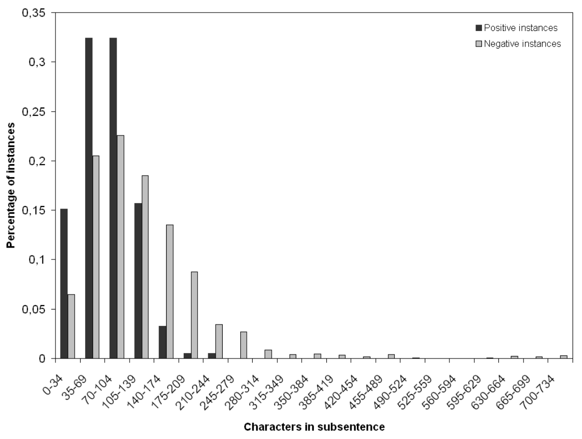


Figure 2: Distribution of Multiple binding instances, according to the length of the sub-sentence (training data).

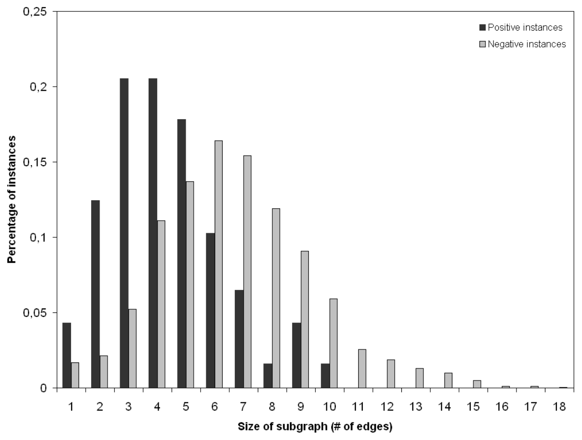


Figure 3: Distribution of Multiple binding instances, according to the size of the subgraph (training data).

Furthermore, for each instance, a minimal subgraph of the dependency graph was extracted, containing the full trigger and all arguments. The size of this subgraph was also used as a parameter for the NI filter, as positive instances are usually expressed in a smaller subtree than negative examples. In Figure 3 we see how the subgraphs of positive Multiple binding instances are never larger than 10 edges, while negative instances can contain up to 18 edges. In this case, only keeping instances with subgraphs smaller than 8 edges will discard many irrelevant negatives, while keeping most of the positive instances.

The NI filter further reduces noise in the data and unbalancedness. We now end up with 4070 Single binding instances (of which 13% positives) and 2365 Multiple binding instances (8% positives). Table 1 shows the final distribution of instances for all event types. Transcription, Localization and Multiple binding have the lowest percentage of positive instances, ranging between 7% and 8%, while Phosphorylation has up to 48% positive instances. It should be noted that the number of positive instances in Table 1 is lower than the actual number of positive examples in the training set, due to limitations of our instance definition method. However, a study regarding maximal recall shows that we do not remove too many true positives (TPs) (more details in Section 4.1).

3.4 Feature generation

For feature generation, we base our method on the rich feature set we previously used in our work on protein-protein interactions (Van Landeghem et al., 2008). The goal of that study was to extract binary relations and only one path in the dependency graph was analyzed for each instance. In the present work however, we are processing larger and more complex subgraphs. This is why we have excluded 'edge walks', i.e. patterns of two consecutive edges and their common vertex (e.g. 'nsubj VBZ prep'). To compensate for the loss of information, we have added trigrams to the feature set. These are three stemmed consecutive words from the sub-sentence spanning the event, e.g. 'by induc transcript', which is the stemmed variant of 'by inducing transcription'. Other features include

- A BOW-approach by looking at all the words which appear at a vertex of the subgraph. This automatically excludes uninformative words such as prepositions.
- Lexical and syntactic information of triggers.
- Size of the subgraph.

Event type	**# neg. inst.**	**# pos. inst.**	**% pos. inst.**
Localization	3415	249	7
Single binding	3548	522	13
Multiple binding	2180	185	8
Gene expression	5356	1542	22
Transcription	6930	489	7
Protein catabolism	175	96	35
Phosphorylation	163	153	48

Table 1: Distribution of instances

Event type	Features
Localization	18 121
Single binding	21 332
Multiple binding	11 228
Gene expression	31 332
Transcription	30 306
Protein catabolism	1 883
Phosphorylation	2 185

Table 2: Dimensionality of the datasets

- Length of the sub-sentence.
- Extra features for Regulation events, storing whether the arguments are proteins or events, specifying the exact event type.
- Vertex walks which consist of two vertices and their connecting edge. For these patterns, both lexical as well as syntactic information is kept. When using lexical information, protein names and triggers were blinded in order to extract more general patterns (e.g. 'trigger nsubj protx' which expresses that the given protein is the subject of a trigger). Blinding avoids overfitting of the classifier.

In the training phase, each instance generates different patterns, and each pattern is stored as a numeric feature in the feature vector. During testing, we count how many times each feature is found for each instance. This results in very sparse and high-dimensional datasets. Table 2 shows the dimensionality of the datasets for all event types. Protein catabolism has the lowest dimensionality with 1883 features, while Transcription and Gene expression produce over 30 000 features.

3.5 Classification

To process our dataset, we had to find a classifier able to deal with thousands of instances, thousands of features, and an unbalancedness of up to 93% negative instances. We have used the LibSVM implementation as provided by WEKA[2], as a few preliminary tests using different classifiers (such as Random Forests) gave worse results. We integrated an internal 5-fold cross-validation loop on the training portion of the data to determine a useful C-parameter. All other parameters were left unchanged, including the type of kernel which is a radial basis function by default.

[2]Available at `http://www.cs.waikato.ac.nz/ml/weka/`

In combination with the LibSVM, we have tried applying feature selection (FS). At first sight, FS did not seem to lead to gain in performance, although we were not able to test this hypothesis more thoroughly due to time limitations of the task. Finally, we have also tested the influence of assigning higher weights to positive training instances, in order to make up for the unbalanced nature of the data, but this had almost no effect on overall performance.

3.6 Post-processing

We have implemented a few custom-made post-processing modules, designed to further reduce FPs and improve precision of our method. We report here on their influence on performance.

Overlapping triggers of different event types

Predictions for different event types were processed in parallel and merged afterwards. This means that two triggers of different event types might overlap, based on the same words in the text. However, a word in natural language can only have one meaning at a time. When two such triggers lead to events with different event types, this means that some of these events should be FPs. When testing on the development data, we found a few predictions where this problem occurred. For example, the trigger 'expression' can lead to both a Transcription and a Gene expression event, but not at the same time. In such a case, we only select the prediction with the highest SVM score. However, thanks to careful construction of the dictionaries (Section 3.2), their mutual overlap is rather small, and thus this post-processing module has almost no influence on performance.

Events based on the same trigger

One trigger might be involved in different events from the same event type. For example, the sentence 'it induces expression of STAT5-regulated genes in CTLL-2, i.e. beta-casein, and oncostatin M (OSM)' mentions two Gene expression events based on the trigger 'expression', one involving beta-casein, and one involving OSM. For these two events, the subgraphs will be very similar, resulting in similar features and SVM scores. However, often a trigger only leads to one true event, while all other candidates from the same event type are false positives. We have carefully benchmarked this hypothesis, and found that for Protein catabolism and Phosphorylation, we could achieve better performance by only keeping the top-ranked prediction. Up to 5% in F-score could be gained for these events. This is due to the fact that for these two event types, usually only one true event is linked to each trigger.

3.7 Negation

We found that there are three major categories of event negation:

1. A negation construct is found in the close vicinity of the trigger (e.g. 'no', 'failure to').
2. A trigger already expresses negation by itself (e.g. 'non-expressing', 'immobilization').
3. A trigger in a certain sentence expresses both positive as negative events. In this case, the pattern 'but not' is often used (e.g. 'overexpression of Vav, but not SLP-76, augments CD28-induced IL-2 promoter activity').

We have created a custom-made rule-based system to process these three categories. The rules make use of small dictionaries collected from the training data. For rule 1, we checked whether a negation word appears right in front of the trigger. To apply rule 2, we used a list of inherent negative triggers deduced from the training set. For rule 3, we checked whether we could find patterns such as 'but not' or 'whereas', negating only the event involving the protein mentioned right after that pattern.

3.8 Speculation

We identified two major reasons why the description of an event could be regarded as speculation instead of a mere fact. These categories are:

1. Uncertainty: the authors state the interactions or events they are investigating, without knowing the true results (yet). This is often indicated with expressions such as 'we have examined whether (...)'.
2. Hypothesis: authors formulate a hypothesis to try and explain the results of an experiment. Specific speculation words such as 'might' or 'appear to' often occur right before the trigger.

Event type	Maximal recall
Localization	84.91 %
Binding	78.23 %
Gene expression	91.57 %
Transcription	90.24 %
Protein catabolism	100 %
Phosphorylation	95.74 %
Regulation	46.15 %
Positive regulation	39.71 %
Negative regulation	43.88 %
Negation	28.97 %
Speculation	25.26 %

Table 3: Maximal recall for the development data

Similar to detecting negation, we compiled a list of relevant expressions from the training data and have used this to implement a simple rule-based system. For rule 1, we checked the appearance of such an expression in a range of 60 characters before the trigger and up to 60 characters after the trigger. Rule 2 was applied on a smaller range: only 20 characters right before the trigger were scanned.

4 Results

Our final machine learning framework consists of all the modules described in the previous section. To summarize, these design choices were made: automatically compiled dictionaries which were cleaned manually, usage of the NI filter, no weights on positive instances, a LibSVM classifier and no feature selection. We used both post-processing modules, but the second one only for Protein catabolism and Phosphorylation events. The best SVM cut-offs were chosen by determining the best F-score on the development data for each classifier.

4.1 Benchmarking on the development data

Protein events

To evaluate maximal recall of our instance extraction method, we executed an evaluation using an all-true classifier. As can be seen in Table 3, maximal recall is quite high for almost all Protein events, meaning that dictionary coverage is good, our NI filter does not remove too many TPs, and not too many events are expressed across sentences and thus not picked up by our method. Binding and Localization are the only events with less than 90% recall. Due to

Event type	Recall	Precision	F-score
Localization	77.36	91.11	83.67
Binding	45.16	37.21	40.80
Gene expression	70.79	79.94	75.08
Transcription	60.98	75.76	67.57
Protein catabolism	80.95	89.47	85.00
Phosphorylation	68.09	88.89	77.11
Total	62.45	64.40	63.41
Regulation	23.67	41.67	30.19
Positive regulation	21.56	38.00	27.51
Negative regulation	30.10	41.26	34.81
Total	23.63	39.39	29.54
Task 1	41.03	53.50	**46.44**
Negation	15.89	45.95	23.61
Speculation	20.00	26.87	22.93
Total	17.82	33.65	23.30
Task 3	38.77	52.24	**44.51**

Table 4: Final performance of all events for the development data

time constraints, we were not able to test which of our modules leads to false negative (FN) instances.

For each event, we have determined the best classifier cut-offs to achieve maximal F-score. Results of the final performance for the predictions of Protein events on the development data, can be seen in Table 4. For most events, we achieve very high precision, thanks to our careful definition of instances in combination with the NI-filter.

Looking at the F-measures, Transcription, Gene expression and Phosphorylation all perform between 67 and 77%, while Localization and Protein catabolism have an F-score of more than 83%. It becomes clear that Binding is the most difficult event type, with a performance of 41% F. Unfortunately, this group of events contains 44% of all Protein events, greatly influencing total performance. Average performance of predicting Protein events results in 63.41% F.

Regulation

When evaluating the predictions of Regulation events, one has to take into account that the performance greatly depends on the ability of our system to predict Protein events. Indeed, one FN Protein event can lead to multiple FN Regulation events, and the same holds for FPs. Furthermore, we do not try to extract events across sentences, which may lead to more FNs. To study maximal recall of the Regulation events, we have again applied an all-true classifier. Table 3 shows that the highest possible recall of the Regulation events is never above 50%, greatly limiting the performance of our method.

As regulation events can participate in new regulation events, one should run the regulation pipeline repeatedly until no more new events are found. In our experiments, we have found that even the first recursive run did not lead to much better performance, and only a few more Regulation events were found.

Final results are shown in Table 4. With recall being rather low, between 21% and 30%, at least we achieve relatively good precision: around 40% for each of the three regulation types. On average, the F-score is almost 30% for the regulation events, which is significantly lower than the performance of Protein events. On average, we obtain an F-score of 46.44% on the development data for task 1.

Negation and speculation

The performance of this subtask depends heavily on the performance of subtask 1. Again we have applied an all-true classifier to determine maximal recall (Table 3). Less than 30% of the events necessary for task 3 can be found with our setup; all of these FNs are due to FNs in task 1.

Final results are shown in Table 4. Performance of around 23% F-score is achieved on the development data. We take into consideration that according to the maximal recall study, only 29% of the necessary events for Negation were extracted by task 1. In the final results, 16% of all the negation events were found. This means that our rule-based method by itself achieves about 55% recall for Negation. Similarly, the system has a recall of 80% for Speculation when only considering events found in task 1. We conclude that our simple rule-based system performs reasonably well.

4.2 Scoring and ranking on final test set

Finally, our system was applied to the test data. Achieving a global F-score of 40.54% for subtask 1, we obtain a 5th place out of 24 participating teams. For subtask 3 of finding negation and speculation, we obtain a second place with a 37.80% F-score.

Final results for each of the event types are shown in Table 5. As on the development data, we see

Event type	Recall	Precision	F-score
Localization	43.68	78.35	56.09
Binding	38.04	38.60	38.32
Gene expression	59.42	81.56	68.75
Transcription	39.42	60.67	47.79
Protein catabolism	64.29	60.00	62.07
Phosphorylation	56.30	89.41	69.09
Total	50.75	67.24	57.85
Regulation	10.65	22.79	14.52
Positive regulation	17.19	32.19	22.41
Negative regulation	22.96	35.22	27.80
Total	17.36	31.61	22.41
Task 1	33.41	51.55	**40.54**
Negation	10.57	45.10	17.13
Speculation	8.65	15.79	11.18
Total	9.66	24.85	13.91
Task 3	30.55	49.57	**37.80**

Table 5: Performance of all events for the final test set

that the Binding event performs worst, and the same trend is found when analyzing results of other teams. In general however, we achieve a high precision: 67% for Protein events, 52% on average on subtask 1, and 50% on average on subtask 3. Another trend which is confirmed by other teams, is the fact that predicting Protein events achieves much higher performance than the prediction of Regulation events.

Compared to our results on the development data (Table 4), we notice a drop of performance for the Protein events of about 0.06F. This loss is propagated to the Regulation events and to Negation and Speculation, each also performing about 0.06F worse than on the development data. We believe this drop in performance might be due to overfitting of the system during training. It is difficult to find the best SVM cut-offs to achieve maximal performance. We have tuned these cut-offs on the development data, but they might not be ideal for the final test set. For this reason, we believe that it might be more representative to use evaluation schemes such as the area under the receiver operating characteristics curve (AUC) measure (Hanley and McNeil, 1982; Airola et al., 2008).

5 Conclusions and future work

We have participated in the BioNLP'09 Shared Task, joining the rest of the community in the progression of relation-based extraction towards the extraction of events from bio-molecular texts. Out of the 24 participants, we see quite some teams with a very good performance, with the highest result achieving an F-score of nearly 52%. We believe the community is off to a good start in this task, and we hope work in this field will continue afterwards.

In our own study, we notice that the task of extracting bio-molecular events leads to high-dimensional and unbalanced datasets. We carefully designed our system in order to improve balance of the datasets and to avoid false positives. For feature generation, we have made use of a modified bag-of-words approach, included trigrams extracted from the sentence, and derived patterns from dependency graphs. Our high-precision framework achieves a fifth position out of 24 participating teams in subtask 1, and second position out of six for subtask 3.

In the future, we would like to investigate the use of feature selection to produce better models for the classification task. Another interesting topic would be how to combine coreference resolution with dependency graphs in order to process events which span multiple sentences in text.

For the community as a whole, we think the next step would be to work on full articles instead of mere abstracts. Also, it might be interesting to investigate the use of text-bound annotation which is not necessarily contiguous, such as is the case in the Bioinfer corpus (Pyysalo et al., 2007), to be able to fully capture the semantics of a certain event.

Acknowledgments

SVL and YS would like to thank the Research Foundation Flanders (FWO) for funding their research. Furthermore, the authors would like to thank the organizers of the BioNLP'09 Shared Task for offering to the community a very valuable and well organized task about event extraction. We believe careful evaluation and discussion of the results will lead to a significant step forward in this domain.

References

A. Airola, S. Pyysalo, J. Björne, T. Pahikkala, F. Ginter and T. Salakoski. 2008. All-paths graph kernel for protein-protein interaction extraction with evaluation of cross-corpus learning. *BMC Bioinformatics*, 9(Suppl 11):S2

R.B. Altman, C.M. Bergman, J. Blake, C. Blaschke, A. Cohen, F. Gannon, L. Grivell, U. Hahn, W. Hersh, L. Hirschman, L.J. Jensen, M. Krallinger, B. Mons, S.I. O'Donoghue, M.C. Peitsch, D. Rebholz-Schuhmann, H. Shatkay and A. Valencia. 2008. Text mining for biology - the way forward: opinions from leading scientists. *Genome Biology*, 9(Suppl 2):S7

B. Boser, I. Guyon and V.N. Vapnik. 1992. A training algorithm for optimal margin classifiers. *Proceedings of the 5th annual workshop on Computational learning theory (COLT)*, 144-152

K. Fundel, R. Küffner and R. Zimmer. 2007. RelEx—Relation extraction using dependency parse trees. *Bioinformatics*, 23(3):365-371

C. Giuliano, A. Lavelli and L. Romano 2006. Exploiting shallow linguistic information for relation extraction from biomedical literature. *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, 401-408

J. Hanley and B. J. McNeil. 1982. The meaning and use of the area under a receiver operating characteristic (roc) curve. *Radiology*, 143(1):29-36

L. Hirschman, A. Yeh, C. Blaschke and A. Valencia. 2005. Overview of BioCreAtIvE: critical assessment of information extraction for biology. *BMC Bioinformatics*, 6(Suppl 1):S1

J.-D. Kim, T. Ohta and J. Tsujii. 2008. Corpus annotation for mining biomedical events from literature. *BMC Bioinformatics*, 19(Suppl 1):i180-i182

J.-D. Kim, T. Ohta, S. Pyssalo, Y. Kano and J. Tsujii. 2009. Overview of BioNLP'09 Shared Task on Event Extraction, *Proceedings of Natural Language Processing in Biomedicine (BioNLP) NAACL 2009 Workshop*, to appear

M. Krallinger, A. Morgan, L. Smith, F. Leitner, L. Tanabe, J. Wilbur, L. Hirschman and A. Valencia. 2008. Evaluation of text-mining systems for biology: overview of the Second BioCreative community challenge. *Genome Biology*, 9(Suppl 2):S1

MC. de Marneffe, B. MacCartney and C.D. Manning. 2006. Generating typed dependency parses from phrase structure parses. *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC)*, 449-454

C. Plake, J. Hakenberg and U. Leser. 2005. Optimizing syntax patterns for discovering protein-protein interactions. *Proceedings of the 2005 ACM symposium on Applied computing (SAC)*, 195-201

M.F. Porter. 1980. An algorithm for suffix stripping. *Program*, 14(3), 130-137

S. Pyysalo, F. Ginter, J. Heimonen, J. Björne, J. Boberg, J. Järvinen and T. Salakoski. 2007. BioInfer: A corpus for information extraction in the biomedical domain. *BMC Bioinformatics*, 8(50)

R. Saetre, K. Sagae and J. Tsujii. 2008. Syntactic features for protein-protein interaction extraction. *Proceedings of the 2nd International Symposium on Languages in Biology and Medicine (LBM)*, 6.1-6.14

S. Van Landeghem, Y. Saeys, B. De Baets and Y. Van de Peer. 2008. Extracting protein-protein interactions from text using rich feature vectors and feature selection. *Proceedings of the Third International Symposium on Semantic Mining in Biomedicine (SMBM)*, 77-84.

Exploring ways beyond the simple supervised learning approach for biological event extraction

György Móra[1], Richárd Farkas[1], György Szarvas[2*], Zsolt Molnár[3]
gymora@gmail.com, rfarkas@inf.u-szeged.hu,
szarvas@tk.informatik.tu-darmstadt.de, zsolt@acheuron.hu
[1] Hungarian Academy of Sciences, Research Group on Artificial Intelligence
Aradi vértanuk tere 1., H-6720 Szeged, Hungary
[2] Ubiquitous Knowledge Processing Lab, Technische Universität Darmstadt
Hochschulstraße 10., D-64289 Darmstadt, Germany
[3] Acheuron Hungary Ltd., Chemo-, and Bioinformatics group,
Tiszavirág u. 11., H-6726 Szeged, Hungary

Abstract

Our paper presents the comparison of a machine-learnt and a manually constructed expert-rule-based biological event extraction system and some preliminary experiments to apply a negation and speculation detection system to further classify the extracted events. We report results on the *BioNLP'09 Shared Task on Event Extraction* evaluation datasets, and also on an external dataset for negation and speculation detection.

1 Introduction

When we consider the sizes of publicly available biomedical scientific literature databases for researchers, valuable biological knowledge is accessible today in enormous amounts. The efficient processing of these large text collections is becoming an increasingly important issue in Natural Language Processing. For a survey on techniques used in biological Information Extraction, see (Tuncbag et al., 2009).

The *BioNLP'09 Shared Task* (Kim et al., 2009) involved the recognition of bio-molecular events in scientific abstracts. In this paper we describe our systems submitted to the *event detection and characterization* (Task1) and the *recognition of negations and speculations* (Task3) subtasks. Our experiments can be regarded as case studies on i) how to define a framework for a hybrid human-machine biological information extraction system, ii) how the linguistic scopes of negation/speculation keywords relate to biological event annotations.

*On leave from RGAI of Hungarian Acad. Sci.

2 Event detection

We formulated the event extraction task as a classification problem for each event-trigger-word/protein pair. A domain expert collected 140 keywords which he found meaningful and reliable by manual inspection of the corpus. This set of high-precision keywords covered 69.8% of the event annotations in the training data.

We analysed each occurrence of these keywords in two different approaches. We used C4.5 decision tree classifier to predict one of the event types considered in the shared task or the keyword/protein pair being unrelated; and we also developed a hand-crafted expert system with a biological expert. We observed that the two systems extract markedly different sets of true positive events. Our final submission was thus the union of the events extracted by the expert-rule-based and the statistical systems (we call this *hybrid system* later on).

2.1 The statistical event classifier

The preprocessing of the data was performed using the *UltraCompare* (Kano et al., 2008) repository provided by the organizers of the challenge: *Genia sentence splitter*, *Genia tagger* for POS coding and NER.

The statistical system classified each keyword/protein pair into 9 event and 2 non-event classes. A pair was either labeled according to the predicted event type (the keyword as an event trigger and the protein name as the theme of the event), `non-event` (keyword not an event trigger) or `wrong-protein` (the theme of the event is a different protein). We chose to use two non-event classes to make the decision tree more human readable (the negative cases being separated). This made the comparison of the statistical model and the rule-based system easier.

The features we used were the following: 1) the words and POS codes in a window (± 3 tokens) around the keyword, preserving position information relative to the keyword; 2) the distances between the keyword and the two nearest annotated proteins (left and right) and the theme candidate as numeric features[1]. The protein annotations were replaced by the term `$protein`, *Genia tagger* annotations by `$genia-protein` (mainly complexes), to enable the classifier to learn the difference between events involved in the shared task, and events out of the scope of the task. Events with protein complexes and families often had the same linguistic structure as events with annotated proteins. As complexes did not form events in the shared task, they sometimes misled our local-context-based classifier. For example '*the binding of **ISGF3***' was not annotated as an event because the theme is not a "protein" (as defined by the shared task guidelines), while '*the binding of **TRAF2***' was (TRAF2 being a protein, and not a complex as in the former example).

We trained a *C4.5* decision tree classifier using *Weka* (Witten and Frank, 2005). The human readable models and fast training time motivated our selection of a learning algorithm which allowed a straightforward comparison with the expert system.

2.2 Expert-rule-based system

The expert system was constructed by a biologist who had over 4 years of experience in similar tasks. The main idea was to define rules – which have a very high precision – in order to compare them with the learnt decision trees and to increase the coverage of the final system by adding these annotations to the output of the statistical system. We only managed to prepare expert rules for the *Phosphorylation* and *Gene_expression* classes due to time constraints (a total of 46 patterns). The expert was asked to construct high-precision rules (they were tested on the train set to keep the false positive rate near zero) in order to gain insight into the structure of reliable rules.

Here each rule is bound to a specific keyword. Every rule is a sequence of "word patterns" (with or without a suffix). A word pattern can match a protein, an arbitrary word, an exact word or the keyword. Every pattern can have a *Regular Expression* style suffix:

Table 1: Word pattern types and suffixes

`<keyword>`	matching the keyword of the event
`"word"`	matching regular words
`_`	matching any token
`$protein`	matching any annotated protein
`?`	zero or one of the word pattern
`*`	zero or more of the word pattern
`+`	one or more of the word pattern
`{a,b}`	definite number of word patterns

For example the `'<expression> _? "of" _? $protein'` pattern recognizes an event with the keyword *expression*, followed by an arbitrary word and then the word *of*, or immediately by *of* and then a protein (or immediately by the protein name).

An obvious drawback of this system is that negation is not allowed, so the expert was unable to define a word pattern like `!"of"` to match any token besides *of*. This extension would have been a straightforward way of improving the system.

2.3 Experimental results

We expected the recall of the hybrid system to be near the sum of the recalls of the individual systems, meaning that they had recognized different events, as the pattern matching was mainly based on the order of the tokens, while the statistical classifier learned position-oriented contextual clues. Thanks to the high precision of the rule-based system, the overall precision also increased. The two event classes which were included in the expert system had a significantly better precision score. The coverage of the *Phosphorylation* class was lower than that for the *Gene_expression* class because its patterns were still incomplete[2].

[1]More information on the features and parameters used can be found at `www.inf.u-szeged.hu/rgai/BioEventExtraction`

[2]A discussion on comparing the contribution of the two approaches and individual rules can be found at `www.inf.u-szeged.hu/rgai/BioEventExtraction`

Table 2: Results of rule based-system compared to the statistical and combined systems (R/P/fscore)

	All Event	Gene_exp.	Phosph.
stat.	16 / 31 / 21	36 / 41 / 38	73 / 37 / 49
rule	5 / 80 / 10	20 / 85 / 33	17 / 58 / 26
hybrid	22 / 37 / 27	56 / 51 / 54	81 / 40 / 53

3 Recognition of negations and speculations

For negation and speculation detection, we applied a model trained on a different dataset (Vincze et al., 2008) of scientific abstracts, which had been specially annotated for negative and uncertain keywords and their linguistic scope. Due to time constraints we used our model to produce annotations for Task3 without any sort of fine tuning to the shared task gold standard annotations.

The only exception here was a subclass of speculative annotations that were not triggered by a word used to express uncertainty, but were judged to be speculative because the sentence itself reported on some experiments performed, the focus of the investigations described in the article, etc. That is, it was not the meaning of the text that was uncertain, but – as saying that something has been examined does not mean it actually exists – the sentence implicitly contained uncertain information. Since such sentences were not covered by our corpus, for these cases we collected the most reliable text cues from the shared task training data and applied a dictionary-lookup-based approach. We did this so as to get a comprehensive model for the Genia negation and speculation task.

As for the explicit uncertain and negative statements, we applied a more sophisticated approach that exploited the annotations of the *BioScope* corpus (Vincze et al., 2008). For each frequent and ambiguous keyword found in the approximately 1200 abstracts annotated in *BioScope*, we trained a separate classifier to discriminate keyword/non-keyword uses of each term, using local contextual patterns (neighbouring lemmas, their POS codes, etc.) as features. In others words, for the most common uncertain and negative keywords, we attempted a context-based disambiguation, instead of a simple keyword lookup. Having the keywords, we predicted their scope using simple heuristics ('*to the end of the sentence*', '*to the next punctation mark in both directions*', etc.). In the shared task we examined each extracted event and they were said to be negated or hedged when some of their arguments (trigger word, theme or clause) were within a linguistic scope.

3.1 Experimental results

First we evaluated our negation and speculation keyword/non-keyword classification models on the *BioScope* corpus by 5-fold cross-validation. We trained models for 15 negation and 41 speculative keywords. We considered different word forms of the same lemma to be different keywords because they may be used in a different meaning/context. For instance, different keyword/non-keyword decision rules must be used for *appear*, *appears* and *appeared*. We trained a *C4.5* decision tree using word uni- and bigram features and POS codes to discriminate keyword/non-keyword uses and compared the results with the most frequent class (MFC) baseline.

Overall, our context-based classification method outperformed the baseline algorithm by 3.7% (giving an error reduction of 46%) and 3.1% (giving an error reduction of 27%) on the negation and speculation keywords, respectively. The learnt models were typically very small decision trees i.e. they represented very simple rules indicating collocations (like '*hypothesis* is a keyword if and only if followed by *that*, etc.). More complex rules (e.g. '*clear* is a keyword if and only if *not* is in ± 3 environment') were learnt just in a few cases.

Our second set of experiments focused on Task3 of the shared task (Kim et al., 2009). As the official evaluation process of Task3 was built upon the detected events of Task1, it did not provide any useful feedback about our negation and speculation detection approach. Thus instead of our Task1 output, we evaluated our model on the gold standard Task1 annotation of the training and the development datasets. The statistical parts of the system were learnt on the *BioScope* corpus, thus the train set was kept blind as well. Table 3 summarises the results obtained by the explicit negation, speculation and by the full speculation (both explicit and implicit keywords) detection methods.

Analysing the errors of the system, we found that

Table 3: Negation and speculation detection results

	Train (R/P/F)	Dev. (R/P/F)
negation	46.9 / 61.3 / 52.8	42.8 / 57.9 / 49.2
exp. spec.	15.4 / 39.5 / 23.6	15.4 / 32.6 / 20.1
full spec.	25.5 / 71.1 / 37.5	27.9 / 65.3 / 39.1

most of the false positives came from the different approaches of the *BioScope* and the *Genia* annotations (see below for a detailed discussion). Most of the false negative predictions were a consequence of the incompleteness of our keyword list.

3.2 Discussion

We applied this negation and speculation detection model more as a case study to assess the usability of the *BioScope* corpus. This means that we did not fine-tune the system to the *Genia* annotations. Our experiments revealed some fundamental and interesting differences between the Genia-interpretation of negation and speculation, and the corpus used by us. The chief difference is that the *BioScope* corpus was constructed following more linguistic-oriented principles than the Genia negation and speculation annotation did, which sought to extract biological information. These differences taken together explain the relatively poor results we got for the shared task.

There are significant differences in the interpretation of both at the keyword level (i.e. what triggers negation/uncertainty and what does not) and in the definition of the scope of keywords. For example, in a sentence like '*have **NO** effect on the inducibility of the IL-2 promoter*', Genia annotation just considers the *effect* to be negated. This means that the *inducibility* of *IL-2* is regarded as an assertive event here. In *BioScope*, the complements of *effect* are also placed within the scope of *no*, thus it would also be annotated as a negative one. We argue here that the above example is not a regular sentence to express the fact: *IL-2* is *inducible*. We rather think that if the paper has some result (evidence) regarding this event, it should be stated elsewhere in the text, and we should not retrieve this information as a fact just based on the above sentence. Thus we argue that more sophisticated guidelines are needed for the consistent annotation and efficient handling of negation and uncertainty in biomedical text mining.

4 Conclusions

We described preliminary experiments on two different approaches which take us beyond the "take-goldstandard-data, extract-some-features, train-a-classifier" approach for biomedical event extraction from scientific texts (incorporating rule-based systems and linguistic negation/uncertainty detection). The systems introduced here participated in the Genia Event annotation shared task. They achieved relatively poor results on this dataset, mainly due to 1) the special annotation guidelines of the shared task (like disregarding events with protein complex or family arguments, and treating subevents as assertive information) and 2) the limited resources we had to allocate for the task during the challenge timeline. We consider that the lessons learnt here are still useful and we also plan to improve our system in the near future.

5 Acknowledgements

The authors would like to thank the organizers of the shared task for their efforts. This work was supported in part by the NKTH grant of the Hungarian government (codename BELAMI).

References

Y. Kano, N. Nguyen, R. Saetre, K. Yoshida, Y. Miyao, Y. Tsuruoka, Y. Matsubayashi, S. Ananiadou, and J. Tsujii. 2008. Filling the gaps between tools and users: a tool comparator, using protein-protein interaction as an example. *Pac Symp Biocomput*.

J-D. Kim, T. Ohta, S. Pyysalo, Y. Kano, and J. Tsujii. 2009. Overview of bionlp'09 shared task on event extraction. In *Proceedings of Natural Language Processing in Biomedicine (BioNLP) NAACL 2009 Workshop*. To appear.

N. Tuncbag, G. Kar, O. Keskin, A. Gursoy, and R. Nussinov. 2009. A survey of available tools and web servers for analysis of protein-protein interactions and interfaces. *Briefings in Bioinformatics*.

V. Vincze, Gy. Szarvas, R. Farkas, Gy. Móra, and J. Csirik. 2008. The bioscope corpus: biomedical texts annotated for uncertainty, negation and their scopes. *BMC Bioinformatics*, 9(Suppl 11):S9.

I. H. Witten and E. Frank. 2005. *Data Mining: Practical Machine Learning Tools and Techniques, Second Edition*. Morgan Kaufmann.

Author Index