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), 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]

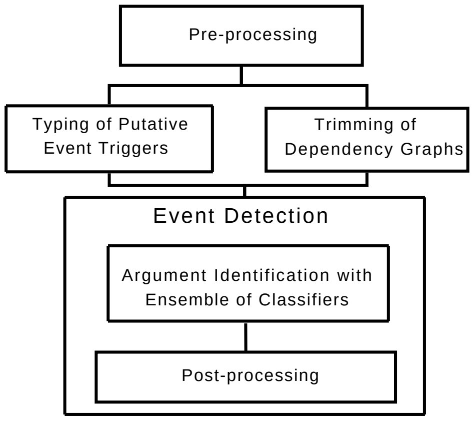
Pre-processing
Typing of Putative Event Triggers
Trimming of Dependency Graphs
Event Detection
Argument Identification with Ensemble of Classifiers
Post-processing

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

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*".

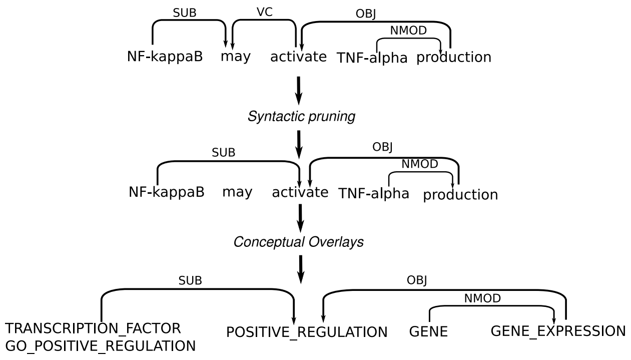

Figure 2: Trimming of Dependency Graphs.

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).

[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`

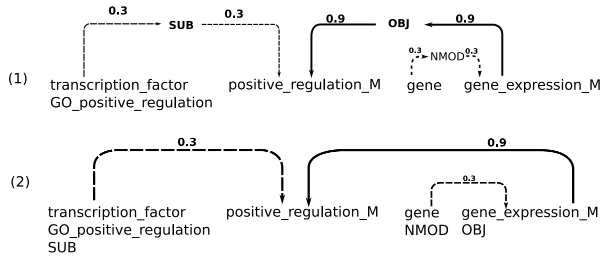

Figure 3: Graph Kernel Representation for a Trimmed Dependency Graph — (1) original representation, (2) representation without graph dependency edge nodes (weights (0.9, 0.3) taken from Airola et al. (2008)).

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.

[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").

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.

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.

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.