Computational Intelligence, Volume xx, Number 000, 2009

Molecular event extraction from Link Grammar parse trees in the BioNLP'09 Shared Task

Võ Há Nguyên, Jörg Hakenberg, Luis Tari, Chitta Baral,
Arizona State University, Tempe, Arizona 85281, USA

Illés Solt,
Budapest University of Technology and Economics, 1117 Budapest, Hungary

Domonkos Tikk, Quang Long Nguyen, Ulf Leser
Humbolt-Universität zu Berlin, 10099 Berlin, Germany

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 the BioNLP Shared Task test data, we achieve an overall F1-measure of 32, 29, and 30% for tasks 1, 2, and 3, respectively.

Key words: text mining, event extraction, sentence parsing, bioinformatics

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

Address correspondence to Jörg Hakenberg at joerg.hakenberg@asu.edu

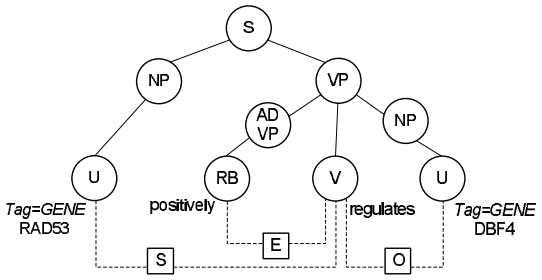

FIGURE 1. Parse tree where constituents are connected by solid lines, linkages between terminals shown as dotted lines. E: adverb to verb, S: subject to verb, O: verb to object.

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 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.

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. 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`.

1.1. 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. Miyao *et al.* (2009) and Pyysalo et al. 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. 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. 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. 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. 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

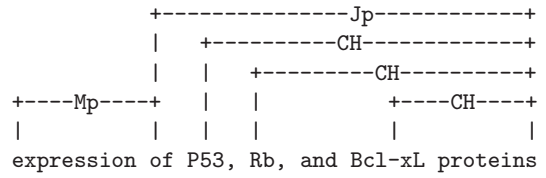

Figure 2. Linkage in a gene expression evidence. Mp: prepositional phrase modifying a noun; Jp: connects preposition to object; CH: noun modifier.

by 45%. Szolovits 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.

2. METHODS

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 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 Bird *et al.* (2006) 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

```
//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.

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 [1]. 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 ≡ {*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

[1] Lucene — see http:// ...

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 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*`⇒`in`⇒`but`→`not`⇒`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

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.

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. 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.

Pair	Query with node variables	Links	Support
Localization	GENE(t1) => localization(e1)	t1→CH→e1	36/237
Binding	GENE(t1) => association(e1)	t1→CH→e1	42/996
Gene expression	GENE(t1) => expression(e1)	t1→CH→e1	347/1507
Transcription	GENE(t1) => gene(a1) => transcription(e1)	t1→CH→a1 and a1→CH→e1	72/498
Protein catab.	proteolysis(e1) => of(a1) => GENE(t1)	e1→M→a1 and a1→J→t1	32/98
Phosphorylation	phosphorylation(e1) => of(a1) => GENE(t1)	e1→M→a1 and a1→J→t1	48/153

Table 2. PTQL queries per argument pair (event↔theme) that have the highest support in the training data. All nodes are bound to variables (round brackets) that are re-used in the *Links* column to depict connections between nodes. Note that event trigger terms given here 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. CH: connects head noun with modifying noun; M: connects nouns to post-nominal modifiers; J: connects prepositions to objects.

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.

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 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 ...*).

Method	TP	FP	FN	P	R	F1
*I*nside trigger	15	8	92	65.2	14.0	23.1
*B*efore trigger	62	17	45	78.5	57.9	66.7
*S*pan-based	6	3	101	66.7	5.6	10.3
*P*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.

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.

*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.

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. DISCUSSION

"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. 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."

"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.'

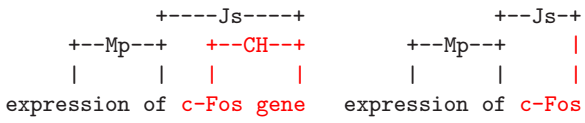

FIGURE 4. Example for alternative structures / optional nodes. In this case, the linkage should reflect the connection from 'expression' to a noun that refers to a gene, independent of its head. The 'Mp' and 'Js' links would be required, the 'CH' link from head to actual gene optional.

4.1. — Extensions to PTQL —

- search for nodes that contain other nodes; example: search for an NP node that contains a gene
- linkages between intermediate constituents, not just leaf nodes; parents (especially NPs) inherit all links from their children
- ⇒ search for an NP that contains a GENE and {the gene or NP} has an MVp link to another node
- include proper POS tags, possible from the Stanford parser, in PTDB

4.2. — Extensions to generating linkage-patterns —

- for pairs given from the training data (protein(s), trigger term, site, etc.), search PubMed for similar sentences and get the linkages from these sentences as additional linkage-patterns
- → parse (a relevant chunk of) PubMed with BioLG, new Banner, etc.
- pre-selection of sentences using a multi-class classifier (set up as 8 one-vs-all classifiers)
- pairwise alignment of linkages (linear encoding) to generate more general patterns (a la YAPPIE)

REFERENCES

Bird, S., Chen, Y., Davidson, S. B., Lee, H., and Zheng, Y. (2006). Designing and evaluating an xpath dialect for linguistic queries. In *ICDE*, page 52, Washington, DC, USA. IEEE Computer Society.

Ding, J., Berleant, D., Xu, J., and Fulmer, A. W. (2003). Extracting biochemical interactions from medline using a link grammar parser. In *5th IEEE International Conference on Tools with Artificial Intelligence*, pages 467–471.

Fundel, K., Küffner, R., and Zimmer, R. (2007). Relex—relation extraction using dependency parse trees. *Bioinformatics*, **23**(3), 365–371.

Hakenberg, J., Plake, C., Royer, L., Strobelt, H., Leser, U., and Schroeder, M. (2008). Gene mention normalization and interaction extraction with context models and sentence motifs. *Genome Biology*, **9**(S2), S14.

Kim, J.-D., Ohta, T., Pyysalo, S., Kano, Y., and Tsujii, J. (2009). Overview of bionlp'09 shared task on event extraction. In *Proceedings of the BioNLP 2009 Workshop Companion Volume for Shared Task*, pages 1–9, Boulder, Colorado. Association for Computational Linguistics.

Kim, S., Yoon, J., and Yang, J. (2008). Kernel approaches for genic interaction extraction. *Bioinformatics*, **24**(1), 118–126.

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

Miyao, Y., Sagae, K., Sætre, R., Matsuzaki, T., and Tsujii, J. (2009). Evaluating contributions of natural language parsers to protein–protein interaction extraction. *Bioinformatics*, **25**(3), 394–400.

Pyysalo, S., Ginter, F., Pahikkala, T., Boberg, J., Järvinen, J., Salakoski, T., and Koivula, J. (2004). Analysis of link grammar on biomedical dependency corpus targeted at protein-protein interactions. In N. Collier, P. Ruch, and A. Nazarenko, editors, *COLING 2004 International Joint workshop on Natural Language Processing in Biomedicine and its Applications (NLPBA/BioNLP) 2004*, pages 15–21, Geneva, Switzerland. COLING.

Pyysalo, S., Salakoski, T., Aubin?, S., and Nazarenko, A. (2006). Lexical adaptation of link grammar to the biomedical sublanguage: a comparative evaluation of three approaches. *BMC*

Bioinformatics, **7**(Suppl 3), S2.

Pyysalo, S., Airola, A., Heimonen, J., Bjorne, J., Ginter, F., and Salakoski, T. (2008). Comparative analysis of five protein-protein interaction corpora. *BMC Bioinformatics*, **9 Suppl 3**, S6.

Sleator, D. and Temperley, D. (1993). Parsing english with a link grammar. In *Third International Workshop on Parsing Technologies*, Tilburg, NL, and Durbuy, B.

Solt, I., Tikk, D., Gál, V., and Kardkovács, Z. T. (2009). Semantic classification of diseases in discharge summaries using a context-aware rule based classifier. *J Am Med Inform Assoc*, **16**(4 – i2b2 Obesity NLP Challenge), 580–584.

Szolovits, P. (2003). Adding a medical lexicon to an english parser. In *AMIA Annu Symp Proc*, pages 639–643, Washington DC, USA.

Tu, P. H., Baral, C., Chen, Y., and Gonzalez, G. (2008). Generalized text extraction from molecular biology text using parse tree database querying. Technical Report TR-08-004, Department of Computer Science and Engineering Arizona State University, Tempe, Arizona, USA.