Computational Intelligence, Volume xx, Number 000, 2009

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

Jörg Hakenberg[1], Illés Solt[2], Domonkos Tikk[2,3], Võ Há Nguyên[1], Luis Tari[1], Quang Long Nguyen[3], Chitta Baral[1], Ulf Leser[3]

[1]*Arizona State University, Tempe, Arizona 85281, USA*
[2]*Budapest University of Technology and Economics, 1117 Budapest, Hungary*
[3]*Humboldt-Universität zu Berlin, 10099 Berlin, Germany*

The BioNLP'09 Shared Task deals with extracting information on molecular events, such as gene expression and protein localization, from natural language text. Information in this benchmark are given as tuples including protein names, trigger terms for each event, and possible other participants such as bindings sites. We address all three tasks of BioNLP'09: event detection, event enrichment, and recognition of negation and speculation. Our method for the first two tasks is based on a deep parser; we store the parse tree of each sentence in a relational database scheme. From the training data, we collect the dependencies connecting any two relevant terms of a known tuple, that is, the shortest paths linking these two constituents. We encode all such linkages in a query language to retrieve similar linkages from unseen text. For the third task, we rely on a hierarchy of hand-crafted regular expressions to recognize speculation and negated events. In this paper, we added extensions regarding a post-processing step that handles ambiguous event trigger terms, as well as an extension of the query language to relax linkage constraints. On the BioNLP Shared Task test data, we achieve an overall F1-measure of 32, 29, and 30% for the successive Tasks 1, 2, and 3, respectively.

Key words: text mining, event extraction, sentence parsing, parse tree database

1. INTRODUCTION

Biomedical text mining aims at making the wealth of information present in publications available for systematic and automated 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 addressed the problem of extracting nine different types of molecular events (Kim *et al.*, 2009) and thus targeted a problem that was considerably less-well studied than protein-protein interactions. Such molecular events included statements about the expression level of genes, the binding sites of proteins, and the up/down regulation of genes, among others. All events focused on genes/proteins and could involve only a single protein (*e.g.*, for protein catabolism), multiple proteins (*e.g.*, in a binding event), and additional arguments (for instance, phosphorylation site or protein location). The most complex type of event considered in the task were regulations, which may refer to other events (negative regulation of gene expression) and may also include causes as arguments. The task also addressed the problem that experimental findings often are described in a speculative 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 subtasks in BioNLP'09 were: *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 subtasks. Essentially, our system consists of three components: a deep parser, a query language for parse trees, and a set of queries

Please address correspondence to Jörg Hakenberg at joerg.hakenberg@asu.edu

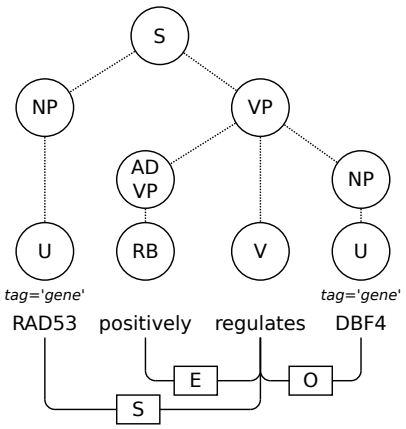

FIGURE 1. Parse tree where constituents are connected by dotted lines, linkages between terminals are shown as solid lines. `E`: adverb to verb, `S`: subject to verb, `O`: verb to object.

that extract specific events from parse trees. First, we use the BioLG parser (Pyysalo *et al.*, 2006) for parsing sentences into a graph 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, called Parse Tree DataBase (PTDB). This information is accessed by a special-purpose query language, called Parse Tree Query Language (PTQL), which matches a user-defined linguistic pattern describing relationships between terms to the database of stored graphs (Tu *et al.*, 2008). 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, and applied to the test data.

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. We call the combination of constituent tree and linkage as produced by the BioLG parser a *parse tree*. Each node in the parse tree has a label and further attributes, such as part-of-speech tags or entity types. Figure 1 shows such a parse tree; dotted lines indicate parent-child relationships in the constituent tree, and solid lines represent the linkages. 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`. Link Grammar uses about 110 of these basic link types (`S`, `O`, `E`), each with various subtypes, providing very rich information on the dependencies between words. In addition to linkages, each leaf node in a parse tree has a `value` (mostly the token itself) and a `tag` attribute that indicates the entity type of a leaf node (such as *gene*). All these information can be used to formulate queries in PTQL, which we will introduce in Section 2.1. PTDB/PTQL are open in a way that further annotations in addition to `value` and `tag` can be added easily; for instance, our general PTDB/PTQL framework supports canonical names and identifiers assigned to nodes (such as UniProt IDs for proteins), both of which can be used in PTQL queries.

In the remainder of this paper, we will first discuss closely related work, before briefly introducing the specific BioNLP'09 Shared Task; for details, we refer the reader to the overview papers by Kim *et al.* (2009). We continue with presenting our methods, starting with an overview of the Parse Tree DataBase (PTDB) and our special-purpose query language PTQL in Section 2. We also explain how we extract PTQL queries from the training data to be used on the test data, and how our systems

deals with task-specific problems such as nested entities. Finally, we describe a post-processing step to deal with ambiguous terms and our solution to handle speculation/negation of events. In Section 3, we present a quantitative analysis resulting from the external evaluation by the Shared Task organizers, as well as a qualitative and error analysis. We conclude with a discussion of the overall method, insights, and future work.

1.1. Related work

We focus our discussion on approaches to information extraction that also use Link Grammar. Evaluations of other deep parsers for information extraction in the life sciences can, for instance, be found in Miyao *et al.* (2009), a survey that assesses different parsers and representations. Fundel *et al.* (2007) and Katrenko and Adriaans (2008) make use of the Stanford Lexicalized Parser in their respective systems to extract protein-protein interactions. The former system, called RelEx, was used again by Pyysalo *et al.* (2008) to compare different protein-protein interaction extraction benchmarks. Note that most other systems based on deep parsing convert information extraction into a classification problem, often using some kind of convolution kernel as in Kim *et al.* (2008); or a local alignment kernel as introduced by Katrenko and Adriaans (2008). Miwa *et al.* (2009) employed a combination of parsers and kernel methods applied to PPI extraction; they achieved an f-score of 72% on the IEPA corpus (Ding *et al.*, 2002). Instead of using classifiers, we employed a pattern-matching approach where patterns were expressed as queries. A similar approach was described in Fundel *et al.* (2007), where three main rules were 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. Overall, we found LinkGrammar/BioLG to be a computationally efficient parser; it provides a rich set of sub-categories of dependencies. Our choice of a query-based approach (resembling linguistic patterns) was motivated by encouraging results of previous experiments (see, for instance, Hao *et al.* (2005); Hakenberg *et al.* (2008); Hunter *et al.* (2008)). Pattern-based approaches tend to yield high precision, as compared to typically high recall of classification-based approaches and co-occurrence plus filtering[1], which was our goal for the BioNLP'09 Shared Task. All three aforementioned pattern-based approaches use information on the token, part-of-speech tag, and entity tag level only; in the work we present here, we also employ information on linkages between words. Furthermore, our goal was to generate a set of such patterns automatically, given a training set such as provided in the Shared Task, instead of hand-crafting these rules.

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.* argued that adaptations to the biomedical domain would enhance the performance. On the IEPA corpus, they reported precision of 65% at 83% recall (f-score 73%), which is comparable to results reported using other methods; the aforementioned RelEx system yielded an f-score of 67% as shown in Pyysalo *et al.* (2008). In another study, 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[2]. 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, as evaluated by Pyysalo *et al.* on two inhouse corpora for protein interaction and transcription; the adaptation 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.

[1]For comparison, sentence-level co-occurrence yields 58% f-score on IEPA, see Pyysalo *et al.* (2008).

[2]Note that recently, the 'standard' Link Grammar parser maintained at `http://www.abisource.com/projects/link-grammar/` was adapted to include the changes made in BioLG.

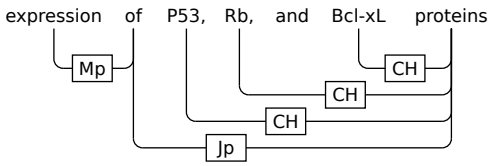

FIGURE 2. Linkage in a gene expression evidence, with 'expression' as event trigger term. `Mp`: prepositional phrase modifying a noun; `Jp`: connects preposition to object; `CH`: noun modifier.

Overall, 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.

1.2. Events and their evaluation in BioNLP'09 Shared Task

The BioNLP'09 Shared Task data focuses on the recognition of nine types of molecular events related to proteins or genes (those are not distinguished). Because the task intends to concentrate on the semantical enrichment of events, protein and gene names are annotated both in the training and test corpora. In Task 1, the recognition of an event consists of the finding the event type, the detection of an event trigger, and the extraction of all primary event arguments. In Task 2, also secondary arguments of the event (see examples below) must be recognized. This may involve some Named Entity Recognition (NER), as the values of these arguments often are not proteins/genes. Different event types have varying numbers of arguments; clearly, the difficulty of event recognition grows proportionally to the number of arguments. The simplest event types (in terms of extraction) are *Gene expression*, *Protein catabolism* and *transcription*, each involving always only one argument (called *theme*). *Phosphorylation* and *Localization* events also have only one primary argument, but may have a secondary argument: *Site* (40% of training examples) in the former, and *AtLoc* (23%) and *ToLoc* (20%) in the latter case. *Binding* events have up to four primary arguments (though 99% of training events have only one or two), and optionally a secondary *Site* argument. *Regulation* events (*Positive, Negative* or without sign) are the most complicated cases, because the primary argument may be an event itself, and may contain any of the *Site, Cause, CSite* secondary arguments.

The evaluation of the task is based on the equality of events: each extracted event is judged either as correct or incorrect as a whole. Evaluation results are reported using the standard information retrieval metrics precision/recall/f-score. Therefore, *incomplete* or *partial recognition* of an event increases both the number of false negatives (complete event not found) and false positives (incomplete event found).

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. Figure 2 shows an example for a linkage in a gene expression event; it illustrates that the event trigger term 'expression' is connected to the three protein themes 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 them into such queries. These components are detailed in Sections 2.1 to 2.5. Section 2.6 explains our methods for context identification regarding negations and speculations. Section 2.7 details the way we handled enumerations.

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 (W3C Consortium, 1999) 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. We have introduced previously a meta-language called PTQL*lite* Tari *et al.* (2009). PTQL*lite* enables non-expert users to write useful queries in a natural language-like style; these queries then get translated automatically into actual PTQL.

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[3]. In the second step, the query is translated into an SQL query, 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. Using a relation database for representation (in our case, MySQL), we can thus benefit from database functionality transparent to our system, such as indexing and query optimization.

```
//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`): `e` ('expression') has to be connected to `a` ('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.

2.2. Extracting PTQL queries

From all events in the training data, we searched for the shortest link paths that connected any two constituents relevant to an event: event triggers to themes, themes to sites, themes to locations, and so on. For each of the different event classes, we obtained at least a set of link paths connecting the event trigger to the theme. Links from themes to sites—required for phosphorylation, binding, and regulation events—were extracted from all such events and then joined into one set; thus, these links may be re-used for all relevant event types, in contrast to the aforementioned links between event triggers and themes. All relevant linkages were transformed into PTQL queries, and we ran these queries against the development and test data sets, respectively. Note that this entire process

[3]Lucene — see http://lucene.apache.org/

of collecting linkages, transformation, and querying, is performed automatically. The automated transformation is a straightforward procedure: *1)* We have to reorder the nodes (tokens, POS) on the identified shortest link path to reflect the original sentence-order (cmp. Figure 3), as the link path might jump back and forth in the sentence. *2)* We assign variables to nodes reflecting sentence order; naming conventions such as `e1` and `e2` for nodes that refer to an event trigger, `t1` referring to a theme, `h1` for head nouns, or `a1` for any other token, help reading/debugging the PTQL queries (notations in Figure 3 are simplified, as there is only one event trigger term, one node referring to the theme, etc.). *3)* Linkages in the PTQL query can be used independent from the sentence order of the involved nodes. As mentioned in the previous section and described in Tu *et al.* (2008), expert user-readable PTQL queries are converted internally into SQL, which is then used to query the relational Parse Tree DataBase scheme.

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. Because many link paths in the training/development data were 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'}

2.3. Searching for genes and proteins nested within noun phrases

An extension to our initial work on the BioNLP'09 Shared Task tackles gene and protein names "obscured" by encapsulating noun phrases. From the training and test data it can be observed that many themes (proteins or genes) occur within noun phrases with differing internal dependencies. Sometimes, the theme itself functions as the head noun, thus most links from outside the phrase will target the theme directly. In other cases, the theme itself will not be the head noun (as in "c-Fos gene", where gene is the head), so the linkages that ultimately should target the theme will differ; see Figure 4 for an example. A PTQL query trying to connect, *e.g.*, an event trigger term to a protein name that functioned as the head noun in a training example will thus fail to extract such differing cases. 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.

These cases attributed to a large proportion of false negatives in our original system (also see Section 3.1). Therefore, for all gene/protein themes, we extended PTQL so we can re-write the queries collected from the training data in the following way. Instead of trying to target a protein theme directly, PTQL is now capable of searching for links targeting any constituent in a noun phrase, instead of just linking leaf nodes in the parse tree (single tokens). We thus look for any noun phrase that contains a protein name, and which also has a constituent with the correct incoming/outgoing dependency. The difference to Section 2.2 and previous descriptions is that now, the protein and the constituent that has the correct link do not necessarily have to be the same, they only need to appear within the same noun phrase. In the example in Figure 4, a query generated from the left or right example also matches the respective other one. Similar special treatments of noun phrases with embedded entities have been discussed in Schneider *et al.* (2009) and Pyysalo *et al.* (2009).

2.4. Post-processing of extracted events

Another extension of the initial system handles a post-processing of events that were extracted by PTQL queries. First, there are many cases where the same event trigger term is ambiguously used for different event types. As an example, the phrase "c-Fos expression" can refer either to a gene expression or a transcription event. The term 'absence' is an event trigger of six different event

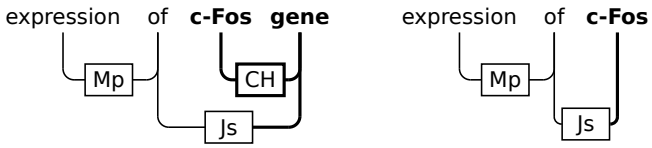

Figure 4. Example for alternative structures / optional nodes. In this case, the linkage should reflect the connection from 'expression' to a noun that refers to a gene, independent of its head. The `Mp` and `Js` links would be required, the `CH` link from head to actual gene optional.

classes in the training set (binding, gene expression, transcription, regulation, positive regulation, as well as negative regulation). Second, in some cases, a sentence might not refer to an event at all, albeit there are one or more matching PTQL queries. Remember that PTQL queries cover only the shortest path between two constituents, and thus ignore any other syntactical/semantic clues in the remainder of the sentence.

To tackle both cases, we run a multi-class classifier obtained from the training data on each predicted event from the test data; the classifier contains all nine event types as individual classes. We use SVM*light* and train it on token and link features: bag-of-word features split into tokens occurring before, between, or after two given constituents (an event trigger term and a protein theme, or a second event term); link features are encoded as *vertex-walks* connecting two tokens ("token–link type–token", where the link type refer to the link connecting the two tokens) or *edge-walks* connecting two links ("link type–token–link type").

2.5. 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 secondary argument. 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 event-class specific prepositions ('of', 'with', 'to', etc.) and determiners between the regulation trigger and the protein or event trigger. Note that a large number of regulation event is embedded into another regulation event as its theme. Treating such cases requires special care, since embedded events 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 sub-expression as the trigger. An example of this approach is:

[[IkappaBalpha]$_{\text{Protein}}$ *induction]*$_{\text{Event:Pos\_reg}}$ *was completely [inhibited]*$_{\text{Trigger:Neg\_reg}}$.

Here, the positive regulation sub-expression is the theme of the negative regulation event triggered by 'inhibited'.

2.6. 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, some-

times 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'

(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*. We derived rules from the training set, *e.g.*, events overlapping with a verb phrase with the verb 'may' are considered speculative. We only kept rules that increased performance on the development set; however, this set contained only 48 abstracts with speculative annotations. 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 applied location-based heuristics by dividing the abstract into title, body, and concluding sentence. In the conclusion part, events were less likely annotated as speculative, if the conclusion also mentioned the word 'results' or its synonyms (*e.g.*, 'data', 'studies', 'observations').

2.7. 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. 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). For the post-processing step alone, as evaluated on the development set, top-k precision is 48.6, 65.0, 70.1, and 81.7% for $k = 1 \ldots 4$, respectively; k refers to finding the correct event class among the first k predictions for each sentence, sorted by their likelihood.

On the BioNLP'09 Shared Task test set, our 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

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	1 507
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	1 845	4 141

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.

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 here, 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. `CH`: connects head noun with modifying noun; `M`: connects nouns to post-nominal modifiers; `J`: connects prepositions to objects.

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 (usage of regular expression for regulation events and enumeration resolution) 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 . . . ").

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 Tasks 2 and 3, approximate span, recursive matching.

Method	TP	FP	FN	P	R	F1
I: inside trigger	15	8	92	65.2	14.0	23.1
B: before trigger	62	17	45	78.5	57.9	66.7
S: span-based	6	3	101	66.7	5.6	10.3
P: parse 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**
$(I \cup B \cup S \cup P) \cup E$	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 benefitial effect of exceptions (E): when permitting all event class–trigger combinations and also NTEs identified as being exceptions, 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-based heuristic	53	47	42	53.0	55.8	54.4
with location-based heuristic	52	34	43	60.5	54.7	**57.5**

TABLE 5. Performance of parse tree based speculation identification, evaluated on the development set. The location-based heuristic identifies the last concluding sentences as affirmative, if they refer to experimental evidence.

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 extremely varied 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). We addressed (3) in this paper, see Section 2.3.

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, Theme 2) 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 30 seconds 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, so that 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.* (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 six event classes, and included the three regulation types with this extended version. For the first six, non-regulation events, we obtain an overall F1-score of 45.6%, for all nine it was 29.3% (Task 2), solving Task 1 implicitly with 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.

In addition to the system presented for the BioNLP Shared Task workshop, we extended the work to include event post-processing (see Section 2.4) to better distinguish between the use of ambiguous trigger terms ('expression', 'absence') on a sentence level. We also extended out query language, PTQL, so that it can address the problem of alternative/optional nodes in an otherwise identical link path (Section 2.3).

In future work, 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, beyond the method discussed in Section 2.3. 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. The idea is to search PubMed for sentences similar to the ones in the training data because they discuss the same terms involved in a known event (same protein, same trigger term, same site, etc.). We would then consider these new sentences as additional training examples.

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.

We are currently working on an extension of PTDB and PTQL to support information from other parsers, thus not being limited to Link Grammar. The overall structure of the database and query syntax would remain identical to the ones shown here and in Tu *et al.* (2008). As a major benefit, users who are more familiar with other parsers can write PTQL queries using parser-specific markup, such as terms referring for dependency types or word categories. We have already extended the PTQL framework to cover the Stanford Lexicalized parser Klein and Manning (2003) and are currently including Enju Miyao and Tsujii (2008), both of which have been used in similar studies in the past. Also, some large document collections have already been parsed using a parser other than Link Grammar; as deep parsing is computationally expensive, it would be beneficary to integrate these data directly. For example, the entire 2009 baseline of Medline has been parsed using the Enju parser and made publicly available [4]. A drawback at first is that PTQL queries written to capture one type of parse information cannot readily be applied to sentences for which parse information was produced by a different parser. Some ideas have been proposed as to the transformation/integration of outputs of different parsers; see, for example, Clegg and Shepherd (2007) and Miyao *et al.* (2009).

REFERENCES

Bird, S., Chen, Y., Davidson, S. B., Lee, H., and Zheng, Y. (2006). Designing and Evaluating an XPath Dialect for Linguistic Queries. In *Proc ICDE*, page 52, Washington, DC, USA. IEEE Computer Society.

Clegg, A. B. and Shepherd, A. J. (2007). Benchmarking natural-language parsers for biological applications using dependency graphs. *BMC Bioinformatics*, **8**, 24.

Ding, J., Berleant, D., Nettleton, D., and Wurtele, E. S. (2002). Mining MEDLINE: Abstracts, Sentences, or Phrases? In *Proc Pacific Symposium on Biocomputing*, pages 326–337, Kaua'i, Hawaii, USA.

Ding, J., Berleant, D., Xu, J., and Fulmer, A. W. (2003). Extracting Biochemical Interactions from MEDLINE Using a Link Grammar Parser. In *IEEE Int Conf 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

[4]See `http://www-tsujii.is.s.u-tokyo.ac.jp/enju-medline/`.

normalization and interaction extraction with context models and sentence motifs. *Genome Biology*, **9**(S2), S14.

Hao, Y., Zhu, X., Huang, M., and Li, M. (2005). Discovering patterns to extract protein-protein interactions from the literature: Part ii. *Bioinformatics*, **21**(15), 3294–3300.

Hunter, L., Lu, Z., Firby, J., Jr, W. A. B., Johnson, H. L., Ogren, P. V., and Cohen, K. B. (2008). OpenDMAP: An open source, ontology-driven concept analysis engine, with applications to capturing knowledge regarding protein transport, protein interactions and cell-type-specific gene expression. *BMC Bioinformatics*, **9**, 78.

Katrenko, S. and Adriaans, P. (2008). A Local Alignment Kernel in the Context of NLP. In *Proc COLING'08*.

Kim, J.-D., Ohta, T., Pyysalo, S., Kano, Y., and Tsujii, J. (2009). Overview of BioNLP'09 Shared Task on Event Extraction. In *Proc BioNLP 2009 Workshop Companion Volume for Shared Task*, pages 1–9, Boulder, Colorado.

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

Klein, D. and Manning, C. D. (2003). Accurate unlexicalized parsing. In *Proc ACL*, pages 423–430.

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.

Miwa, M., Saetre, R., Miyao, Y., and Tsujii, J. (2009). Protein-protein interaction extraction by leveraging multiple kernels and parsers. *Int J Med Inform*.

Miyao, Y. and Tsujii, J. (2008). Feature forest models for probabilistic HPSG parsing. *Computational Linguistics*, **34**, 35–80.

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, *Int Workshop on Natural Language Processing in Biomedicine and its Applications (NLPBA/BioNLP) at COLING 2004*, pages 15–21, Geneva, Switzerland.

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.

Pyysalo, S., Ohta, T., Kim, J., and Tsujii, J. (2009). Static relations: a piece in the biomedical information extraction puzzle. In *Proc Workshop on BioNLP*, pages 1–9. Association for Computational Linguistics.

Schneider, G., Kaljurand, K., and Rinaldi, F. (2009). Detecting protein-protein interactions in biomedical texts using a parser and linguistic resources. In *Proc Computational Linguistics and Intelligent Text Processing (CICLING'09)*, number 5449 in LNCS, pages 406–417. Springer.

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.

Tari, L., Hakenberg, J., Gonzalez, G., and Baral, C. (2009). Querying a parse tree database of Medline text to synthesize user-specific biomolecular networks. In *Proc Pacific Symposium on Biocomputing*, pages 87–98, Kona, Hawaii, 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.

W3C Consortium (1999). XML Path Language (XPath). `http://www.w3.org/TR/xpath`.