The scientific literature is the main source for comprehensive, up-to-date biological knowledge. Automatic extraction of this knowledge facilitates core biological tasks, such as database curation and knowledge discovery. We present here a linguistically inspired, rule-based and syntax-driven methodology for biological event extraction. We rely on a dictionary of trigger words to detect and characterize event expressions and syntactic dependency based heuristics to extract their event arguments. We refine and extend our prior work to recognize speculated and negated events. We show that heuristics based on syntactic dependencies, used to identify event arguments, extend naturally to also identify speculation and negation scope. In the BioNLP’09 Shared Task on Event Extraction, our system placed third in the Core Event Extraction Task (F-score of 0.4462), and first in the Speculation and Negation Task (F-score of 0.4252). Of particular interest is the extraction of complex regulatory events, where it scored second place. Our system significantly outperformed other participating systems in detecting speculation and negation. These results demonstrate the utility of a syntax-driven approach. In this article, we also report on our more recent work on supervised learning of event trigger expressions and discuss event annotation issues, based on our corpus analysis.

Key words: BioNLP, biological event extraction, dependency parsing, heuristic system, negation detection, speculation recognition.

1. INTRODUCTION

The molecular biology literature is growing at an explosive rate, a growth accelerated by an exponential increase in the amount of high throughput genomic data generated by large-scale efforts such as the Human Genome Project. The scientific literature remains the primary medium for disseminating new knowledge in molecular biology and, thus, sophisticated approaches to accessing this massive body of knowledge are necessary. There is a growing awareness among molecular biologists that automated exploitation of the literature is needed (Cohen and Hunter 2004). Natural language processing (NLP) and text mining techniques are expected to help researchers in their tasks by facilitating biological analyses as well as biological database curation. Extracting facts (relations and events) directly from the scientific literature is a crucial, fundamental step in this direction. Advances made in foundational NLP areas, such as parsing and named entity recognition, allow us to focus on biological event extraction (Zweigenbaum et al. 2007). On the other hand, Wilbur, Rzhetsky, and Shatkay (2006) recognize that factual information mining is not sufficient for these tasks and emphasize the need for identifying such phenomena as uncertainty, speculation and polarity.

The difficulty of extracting biological information from scientific literature is due to several factors. First, sentences are long and often have long-range dependencies (Fundel, Küffner, and Zimmer 2007). In addition, the biological processes described are generally complex, involving multiple genes or proteins as well as other biological processes. Furthermore, biological text is rich in higher level linguistic phenomena, such as speculation and negation, which need to be addressed for correct interpretation of the text. Despite all this complexity, however, molecular biology text defines a sublanguage (Friedman et al. 2002), making it amenable to automated approaches. Furthermore, a closer look at various
biological corpora suggests that regularity of this sublanguage may potentially be exploited using relatively simple heuristics.

The major focus of biological information extraction has been on extracting protein–protein interactions, as evinced by two shared tasks, LLL05 (Learning Language in Logic) Challenge (Nédellec 2005) and BioCreAtIvE (Hirschman et al. 2005; Krallinger et al. 2008). The BioNLP’09 Shared Task on Event Extraction (Kim et al. 2009) is more recent and differs from these efforts in several aspects: First, it adopts a more semantically complex definition of bio-events; event type definitions are finer grained, semantic roles of event arguments are considered, and, for certain event types, subevents are allowed as event arguments. Secondly, in contrast to prior efforts, bio-event annotations are bound to textual evidence, minimizing the role of inferencing, context or domain knowledge in annotations. Thirdly, speculation and negation, both higher level linguistic phenomena, are addressed for the first time by a shared task. Finally, by assuming that protein recognition has already been performed, it concentrates on novel aspects of biological event extraction and provides a forum for presentation of more focused efforts.

We present a general and extensible bio-event extraction system. The system development was informed by various aspects of the BioNLP’09 Shared Task on Event Extraction (referred to as the Shared Task, henceforth), which also provides the evaluation platform. We participated in Task 1, core event extraction, and Task 3, negation and speculation recognition, of the Shared Task. Our core event extraction pipeline consists of three phases: event trigger detection, event argument identification and postprocessing. For the competition, we addressed event trigger detection with a simple dictionary-based approach. Following the competition, we also experimented with a more sophisticated, machine learning based approach to assess the pros and cons of each technique. Our event argument identification component is primarily based on a grammar that considers the syntactic dependency paths between event trigger expressions and event arguments. We use simple heuristics in the postprocessing phase to tackle some of the limitations imposed by our approach on event extraction. For speculation and negation recognition (Task 3), we extended and refined our prior work in speculative language recognition (Kilicoglu and Bergler 2008), which involved trigger expressions and dependency relations, as well. The same principles extend naturally to underlie negation detection. Our results show that dependency relations, despite their imperfections, provide a good foundation, on which accurate and reliable event extraction systems can be built and that the regularities of biological text can be adequately exploited with a limited set of syntactic patterns. The results for Task 3, attempted by only six participating groups, confirm our system as the state-of-the-art in this relatively new and experimental research area. Furthermore, our careful analysis of the Shared Task corpus highlights some of the challenges in annotating biological events as well as speculation.

2. BACKGROUND

2.1. Dependency Parsing

Analysis of syntactic dependency relations is at the core of our event extraction system. In dependency parse representation (Mel’čuk 1988; de Marneffe, MacCartney, and Manning 2006), a dependency relation is formalized as a directional grammatical relationship involving two words (head and dependent) and a sentence is represented as a graph of dependency relations. In a typed dependency relation, the type of the grammatical relation between the words, such as nominal subject or direct object, is also specified.
With its ability to reveal long-range syntactic dependencies more directly, dependency parse representation is suitable for building event extraction systems. Dependencies typed with grammatical relations, in particular, benefit such applications. While there is no universally accepted dependency representation, the Stanford dependency scheme (de Marneffe et al. 2006) has been proposed as a common representation for biological applications (Clegg and Shepherd 2007) and is widely used. Björne et al. (2008) argue that Stanford “collapsed” dependency representation, in particular, closely captures the relevant semantics of biomedical text. Stanford dependency parses are derived from underlying phrase structure parses by a module in the public domain, which can be used with any parser that adheres to the Penn Treebank annotations (Bies 1995).

2.2. Biological Relation Extraction

Several approaches to biological relation extraction have been proposed. Simple co-occurrence based approaches (Jenssen et al. 2001; Ding et al. 2002; Rebholz-Schuhmann et al. 2007) assume that biological entities that are mentioned together frequently are related and provide high recall at the expense of low precision. At the other end of the spectrum, shallow parsing and hand-crafted syntactic templates (Blaschke et al. 1999; Rindflesch et al. 2000; Blaschke and Valencia 2001; Friedman et al. 2001; Ono et al. 2001; Leroy, Chen, and Martinez 2003; Saric et al. 2006; Ahlers et al. 2007) as well as full parsing (Daraselia et al. 2004; Yakushijin et al. 2005), provide the basis for relation extraction. In contrast to cooccurrence based methods, these more sophisticated approaches provide higher precision at the expense of lower recall. Approaches combining the strengths of complementary models have also been proposed for high recall and precision (Bunescu et al. 2006).

More recently, dependency parse representation has found considerable use in relation extraction, particularly in extraction of protein–protein interactions. The RelEx system (Fundel et al. 2007) uses Stanford dependency parses of Medline abstracts as the basis for rules that extract gene–protein interactions. Rinaldi et al. (2007) extract relations combining a handwritten grammar based on dependency parsing with a statistical language model. Airola et al. (2008) extract protein–protein interactions from scientific literature using supervised machine learning based on an all-dependency-paths kernel, while Van Landeghem, Saeys, and Van de Peer (2008) use dependency graph-based rich feature vectors for the task and investigate the effect of feature selection. On the other hand, Sætre, Sagae, and Tsujii (2007) augment deep syntactic parsing with a shallow dependency parser to increase the accuracy of the learned model. Finally, Miyao et al. (2009) combine dependency-based features from various syntactic parsers in a supervised machine learning setting, reporting state-of-the-art results in extraction of protein–protein interactions (F-score of 59.5 on AIMED data set).

2.3. Negation and Speculation in Biomedical Text

Recognizing whether extracted events are negated is crucial, as negation often reverses the meaning of a proposition. Most of the work on negation in the biomedical domain has focused on finding negated terms or concepts, particularly within clinical reports. Some of these systems are rule-based and rely on lexical or syntactic information (Chapman et al. 2001; Muralik, Deshpande, and Nadkarni 2001; Elkin et al. 2005; Huang and Lowe 2007; Sanchez-Grau and Poesio 2007); while others (Goldin and Chapman 2003; Averbuch et al. 2004; Rokach, Roman, and Maimon 2008) experiment with machine learning techniques. A recent study, Morante, Liekens, and Daelemans (2008), focuses on learning negation scope using memory-based classifiers trained on the BioScope corpus (Vincze et al. 2008), a data
set in which the scope of negative statements as well as that of speculative statements are annotated.

The speculative aspect of the biomedical literature (also referred to as *hedging*) has been the focus of several recent studies, which deal primarily with distinguishing speculative sentences from nonspeculative ones. Supervised machine learning techniques mostly dominate this area of research (Light, Qiu, and Srinivasan 2004; Medlock and Briscoe 2007; Szarvas 2008), while simple regular expressions have also been explored (Light et al. 2004). Medlock and Briscoe (2007) manually create a hedge classification data set and use weakly supervised learning with an SVM classifier to recognize speculative sentences. Szarvas (2008) extends this work by performing feature selection, using bi- and trigrams and exploiting external dictionaries. We use a more linguistically-oriented approach, relying on a dictionary of weighted lexical hedging cues and syntactic patterns (Kilicoglu and Bergler 2008). The dictionary of hedge triggers is induced semi-automatically. Morante and Daelemans (2009) extend their work on finding the scope of negation and present a machine learning system that finds the scope of hedge cues, again trained on the BioScope corpus.

2.4. Shared Task

The Shared Task aims to provide consistent task definitions, data sets and evaluation for biological information extraction systems based on rich semantics. In this section, we provide some basic details on aspects of the Shared Task relevant to our subsequent discussion.

The Shared Task is divided into three subtasks that focus on different aspects of event extraction. The mandatory Task 1 focuses on *core event extraction*, and involves event trigger detection, event typing and primary argument identification (*Theme* and *Cause*). Task 2 is optional and is concerned with *event enrichment*; that is, recognizing entities other than proteins and assigning these as secondary event arguments (e.g., *Site, toLoc*). The optional Task 3 involves identifying speculated and negated events.

The Shared Task data is based on the GENIA event corpus (Kim, Ohta, and Tsujii 2008). The training and development data sets are derived from the publicly available portion of the corpus, and consist of 800 and 150 Medline abstracts, respectively (8,597 and 1,809 events). On the other hand, the test data set is from a previously unpublished portion of the corpus and consists of 260 Medline abstracts (3,182 events).

The core event types addressed in the Shared Task can be grouped into three classes based on the semantic roles of the arguments involved. In the first group are simple event types involving a single Theme argument (*Gene_expression, Transcription, Protein_catabolism, Phosphorylation and Localization*). The *Binding* event type is in its own class and may involve one or more Theme arguments. Finally, complex regulatory event types (*Regulation, Positive_regulation, Negative_regulation*) may involve a Cause argument, in addition to a Theme. They can also take other events as arguments in either role, making the task of identifying the arguments of such events more challenging. The *Binding* event type, with its variable number of Themes, also presents challenges in event argument identification.

One of the interesting aspects of the shared task is that evaluation is provided at diverse levels and aspects. In the *strict matching* mode, exact equality of event types, triggers and arguments is required. On the other hand, the *approximate span matching* mode relaxes the span matching requirement and allows the extracted spans to be within one word to the left or right of the gold spans. *Approximate recursive matching* considers only the Theme arguments of the subevents for equality. Finally, in *event decomposition* mode, events with
multiple arguments are decomposed into multiple single-argument events and each resulting event is evaluated separately.

Figure 1 shows a sample sentence fragment from the Shared Task data and its gold standard annotations, as well as the corresponding dependency graph. The entity annotations are provided as input, while the participating systems are expected to extract the trigger and event annotations. Throughout this document, we adopt a slightly different representation for events than shown in Figure 1 for better readability. This simplified representation and an event annotation based on it are given below:

(1) a. `<EventType>(<EventID>,Trigger=<TriggerExpression>, Theme=<ProteinOrEventID>, [OptionalArguments])`
   b. `Regulation(E17,Trigger=involved,Theme=E18,Cause=ETS1)`

2.5. Overview

Our approach to event recognition is most similar to the work of Fundel et al. (2007) as it builds on dependency-based heuristics. However, we address a larger number of event types, including complex regulatory events allowing other events as arguments. In addition, event trigger detection plays a more central role in our approach, contrasting with their system and most other PPI systems that rely on finding dependency paths between biological entities of interest. With respect to the role of event triggers, our work is closer to that of Rindflesch et al. (2000) and Saric et al. (2006). We build on prior work on hedging for speculation recognition, extending it to consider negation as well as exploiting dependency relations to a greater extent.
3. EVENT DETECTION AND CHARACTERIZATION

Event recognition assumes that the input text has been preprocessed and appropriate representation has been created. To recognize and fully characterize the event, we proceed to first find event trigger words using a dictionary of categorized event triggers. Then, we heuristically determine salient arguments using Stanford dependency relations. Two postprocessing rules constitute the final event recognition step.

3.1. Preprocessing

Our system requires annotated input, rather than raw text. In each document, sentences, their offset positions in the document and their dependency parses are annotated. Within each sentence, named entities (proteins) and their offset positions as well as word information (tokens, part-of-speech tags, lemmas, and indexes) are marked. For biological event extraction, we combine these annotations, the tokenized input, and dependency annotations in an XML representation. We use the Stanford Lexicalized Parser (Klein and Manning 2003) to extract word-related information, as well as for dependency parsing. The named entities are provided in the Shared Task data.

3.2. Dictionary-Based Approach to Event Triggers

After parsing the training data and creating an enriched document representation, we constructed a dictionary of event triggers. For this purpose, we extracted all event triggers annotated at least twice in the Shared Task training data and refined this set further, based on the principles and generalizations listed below:

(1) **Event triggers are predicates.** We adopt the TimeML (Pustejovsky et al. 2005) notion of event predicates and restrict event triggers to verbs, nouns, and adjectives. Our analysis suggests that trigger words having other part-of-speech tags are tenuously annotated event triggers and, in fact, require either more context or inferencing on the part of the reader to qualify as event triggers (e.g., the preposition by to trigger a Positive_regulation event). Such tenuous trigger expressions constitute 1.2% of all the annotated triggers in the training corpus.

(2) **A single word in a multiword event trigger often dominates the core event meaning.** Multiword event triggers constitute approximately 8% of all triggers in the data. Corpus analysis reveals that core trigger meaning of these multiword event triggers often come from a single word token in the fragment annotated as event trigger. For instance, for trigger transcriptional activation, the annotated event type is Positive_regulation, which suggests that the head of this noun phrase, activation, carries the meaning in this instance (because transcriptional is an event trigger for the Transcription event type, while activation most frequently triggers Positive_regulation events). In other instances, the modifier may carry the trigger meaning if the head word is an “empty head” (Guthrie et al. 1990; Ahlers et al. 2007). For example, the trigger binding activity is annotated as triggering a Binding event, indicating that the head word activity is semantically empty in this context. We note several clear exceptions to this “single word trigger” constraint (e.g., negatively regulate, positive regulation) and address them in the postprocessing step.

(3) **Distribution of event types for a trigger word is generally skewed in favor of a single event type.** Trigger words are ambiguous; that is, they may map to different event types in different contexts. For instance, the nominal trigger induction
appears in the training data 7 times as the trigger for a Gene_expression event, 6 times for a Transcription event, and 169 times for a Positive_regulation event. In addition, it appears 232 times as a nontrigger. Other trigger words, such as presence, are distributed more evenly between event types; however, these cases are less common. We attempt to address the problem of trigger word sense ambiguity by simply imposing a “one event class per word” constraint, motivated by the often skewed distributions of trigger words among event types. With this constraint, induction is taken to trigger Positive_regulation events in all cases.

We refined the remaining event trigger set (303 event triggers) further through limited expansion and filtering, based on several observations:

**Triggers with prefixes** The event trigger set was expanded to include both hyphenated and nonhyphenated forms of event triggers with known prefixes, such as co-, down-, and up- (e.g., coexpression, down-regulate). This resulted in addition of 10 triggers.

**Inflectional/derivational forms** For a trigger that has inflectional/derivational forms acting as triggers in the development data but not in the training data, we added these forms as event triggers. Examples include adding dimerization after dimerize, engage after engagement and the adjective diminished after diminish, among others. The inflectional/derivational forms were confirmed using the UMLS SPECIALIST Lexicon (McCray, Srinivasan, and Browne 1994). We added 19 triggers based on this expansion.

**Tenuous triggers** We manually removed several words from the list of annotated event triggers, which, despite having acceptable part-of-speech tags, still required inferencing to qualify as event triggers for the corresponding event types (e.g., follow for the Positive_regulation type and absence, absent, lack for the Negative_regulation type). 7 such triggers were eliminated from further processing.

**Unreliable triggers** Many words annotated as triggers are in fact overwhelmingly nontriggers in the training corpus. An example is the nominal trigger level, which appears 42 times as a trigger for one of 4 different event types and 455 times as a nontrigger. To eliminate such unreliable event triggers and ensure high precision, we compute a reliability score of predicted event types via maximum likelihood estimation. For a given event type $C$ and event trigger $t$, the reliability score $G(t, C)$ is:

$$ G(t, C) = \frac{w(C : t)}{w(t)} $$

where $w(C : t)$ is the number of times $t$ occurs as a trigger for the event type $C$ and $w(t)$ is the frequency of trigger $t$ in the training corpus. The event triggers added with term expansion are assigned the same scores as the trigger they are derived from. In the subsequent event extraction step, a reliability score threshold is used to eliminate unreliable triggers from consideration.

3.3. Supervised Learning of Event Triggers

Our dictionary-based approach to event triggers is simple and requires little computational effort. However, it is naturally limited to the trigger expressions encountered in the training corpus and a small set of words obtained by expansion. Furthermore, due to the “one event type per word” constraint, it does not allow contextual disambiguation of trigger words. To investigate whether we can address these limitations, we explored supervised learning of event triggers after the shared task competition. We posed the following questions: Can a sophisticated supervised learning method relying on a fairly rich set of features overcome the limitations of a simple and narrow dictionary-based approach? Would the event extraction
The performance obtained with such a supervised approach to event trigger detection justify the complexity and cost of learning this model?

We approach the problem of detecting event triggers as a multiclass classification problem. Rather than training a single multiclass classifier, we take a binary, “one versus the rest” approach to classification and train a binary classifier for each event type. Each training example concerns a potential trigger word, and it is labeled as positive, if the word indicates the corresponding event type and negative otherwise. Only nouns, verbs, and adjectives are considered potential trigger words, as with the dictionary-based approach. In addition, certain words with acceptable part-of-speech tags are not considered as target words. These include all-capitalized words, words ending in digits, words of protein terms and words with single character lemmas (essentially, tokens likely to correspond to biological entities or numbers).

The event type for a target word is predicted to be one corresponding to the classifier that assigns highest confidence score. Further, a confidence score is used to adjust precision versus recall. In other words, if, for a given target word, the highest confidence score comes from the Transcription classifier and if this score is higher than the predetermined threshold, the target word is considered a trigger for a Transcription event.

The feature set we use as input to the binary classifiers consists of the lemma of the target word, the syntactic dependency relations it participates in and whether the word is a nominalization. Dependency relation types are also recorded. All words of protein terms are replaced by the string PROTEIN in dependency relation features, as shown in Example (2), where 5-LOX is a protein.

(2) a. nsubjpass(coexpressed,5-LOX)
b. DEP_nsubjpass_coexpress_PROTEIN

This abstraction mimics limited typing of entities, assuming it would benefit supervised learning. We also experimented with additional features, such as bi- and trigrams, or whether the word appears as a trigger word in the training data; however, we have not observed any increase in performance with these features, and did not include them in the final feature set.

Training data for the classifiers consisted of 92,381 training examples and 71,715 features (6,323 of which are word tokens). No feature selection or weighting was performed. As the supervised learning algorithm, we used the *svmlight* implementation\(^2\) of the Support Vector Machine (SVM) method, known for its ability to handle large feature sets. We optimized SVM classifier parameters (C regularization parameter and J cost factor) by measuring the effect of parameters on the subsequent event extraction performance on the development corpus. As mentioned above, we also adjusted the confidence score threshold (\(T\)) to find the optimum precision and recall. The optimum parameters were found to be: \(C = 0.05\), \(J = 8\), and \(T = -0.75\).

### 3.4. Event Argument Identification

Our syntax-driven approach to event argument identification assumes that the event triggers in the sentence have been identified. It relies mainly on finding dependency paths between these trigger words and the protein terms in the sentence. The trigger word and the types of the dependency relations along the dependency path help to determine whether the protein term is an event argument and what its semantic role is. This procedure is governed

---

1 We find this technique useful when planning on incrementally building a larger system.

by a grammar that we developed based on the “collapsed” dependency output of Stanford Parser (de Marneffe et al. 2006). Complex regulatory events can take other events as well as entity arguments and, therefore, require special attention. For such events, we assume that the semantic head of an event is its trigger word and look for an appropriate dependency path between the main regulatory event trigger and its subevent trigger.

We induced the dependency-based grammar in a semi-automatic manner. We automatically extracted typed dependency relation paths connecting event triggers to corresponding event arguments in the training corpus and ranked them by frequency. We then manually analyzed these paths and the corresponding examples and implemented as grammar rules those deemed to be both correct and sufficiently general. More than 2,000 dependency paths between triggers and arguments were found; however, their distribution was Zipfian, with approximately 70% of them occurring only once. We concentrated on the most frequent, therefore general, dependency paths. Unsurprisingly, the most frequent dependency path involves the dobj (direct object) dependency between verbal event triggers and Theme arguments, occurring 826 times. Next are the nn (nominal modifier) and the prep_of (prepositional modifier headed by of) dependencies between nominal event triggers and their Theme arguments. As expected, the most frequent dependency for Cause arguments is nsubj (nominal subject), followed by the agent (agent) dependency. In Example (3), a prep_of dependency allows us to extract the event given in Example (3c).

(3) a. . . . LPS-specific induction of IL-6 . . .  
   b. prep_of(induction,IL-6)  
   c. Positive_regulation(E1, Trigger = induction, Theme = IL-6)

The ranking of dependency paths indicates that path length is inversely proportional to reliability. In the end, we implemented 27 of these dependency path patterns as grammar rules.

Events allowing event arguments (complex regulatory events) are treated essentially the same way as events taking entity arguments, the main difference being that a dependency path is sought between the trigger of the main event and the trigger of its subevent. In the following example, a Theme dependency path (dobj) is found between two event triggers, mediate and expression, allowing us to extract the first event (E1) as a subevent for the second event (E2).

(4) a. Multiple prolactin-responsive elements mediate G1 and S phase expression of the interferon regulatory factor-1 gene.  
   b. dobj(mediate,expression) prep_of(expression,gene)  
   c. Gene_expression(E1, Trigger=expression, Theme=interferon regulatory factor-1 gene)  
      Positive_regulation(E2, Trigger=mediate, Theme=E1).

Several additional patterns specifically address shortcomings of the Stanford dependency parsing. Prepositional phrases are often attached incorrectly, causing problems in event argument identification. Consider Example (5), where one of the more frequent dependency paths, dobj-prep_on (direct object dependency followed by prepositional modifier headed by on), occurs between the event trigger effect in italic and (expression), a subevent trigger:

(5) a. We have examined the effect of leukotriene B4 (LTB4), a potent lipid proinflammatory mediator, on the expression of the proto-oncogenes c-jun and c-fos.

The ranking of dependency paths indicates that path length is inversely proportional to reliability. In the end, we implemented 27 of these dependency path patterns as grammar rules.

Events allowing event arguments (complex regulatory events) are treated essentially the same way as events taking entity arguments, the main difference being that a dependency path is sought between the trigger of the main event and the trigger of its subevent. In the following example, a Theme dependency path (dobj) is found between two event triggers, mediate and expression, allowing us to extract the first event (E1) as a subevent for the second event (E2).

(4) a. Multiple prolactin-responsive elements mediate G1 and S phase expression of the interferon regulatory factor-1 gene.  
   b. dobj(mediate,expression) prep_of(expression,gene)  
   c. Gene_expression(E1, Trigger=expression, Theme=interferon regulatory factor-1 gene)  
      Positive_regulation(E2, Trigger=mediate, Theme=E1).

Several additional patterns specifically address shortcomings of the Stanford dependency parsing. Prepositional phrases are often attached incorrectly, causing problems in event argument identification. Consider Example (5), where one of the more frequent dependency paths, dobj-prep_on (direct object dependency followed by prepositional modifier headed by on), occurs between the event trigger effect in italic and (expression), a subevent trigger:

(5) a. We have examined the effect of leukotriene B4 (LTB4), a potent lipid proinflammatory mediator, on the expression of the proto-oncogenes c-jun and c-fos.
b. \textit{dobj(examined, effect)}
\textit{∗prep_on(examined, expression)}\(^3\)
\textit{prep_of(expression, c-jun)}
c. \text{Regulation}(E1, \text{Trigger} = \textit{effect}, \text{Theme} = \text{E2})
\text{Gene_expression}
\text{(E2, \text{Trigger} = \textit{expression}, \text{Theme} = \textit{c-jun})}

The second dependency relation points to a prepositional phrase attachment error, where the preposition \textit{on}, which should attach to \textit{effect}, is incorrectly attached to \textit{examined}. If the attachment were correct, the dependency path pattern involving the \textit{prep_on} dependency would allow creating the event \text{E1} in Example (5c). Although the dependency path \textit{dobj-prep_on} is incorrect, it occurs almost exclusively with PP attachment errors involving the preposition \textit{on}, leading us to stipulate it as a “corrective” dependency path and restricting it to work with several trigger words (e.g., \textit{effect}, \textit{influence}, \textit{impact} in this case) whose arguments can be attached by \textit{on}. Postnominal prepositional phrase attachment heuristics detailed in Schuman and Bergler (2006) helped determine six such patterns.

An interesting phenomenon concerns noun phrases with hyphenated adjectival modifiers (e.g., “... \text{LPS-mediated TF expression} ...”), occurring frequently in molecular biology texts and generally involving complex regulatory events. Such cases do not involve a dependency path, as the argument (in this case, \text{LPS}) and the event trigger (\text{mediated}) form a single word. An additional rule addresses these cases, stipulating that the protein term preceding the hyphen is the Cause argument of the regulatory event triggered by the substring following the hyphen. This rule allows us to extract the \text{Positive_regulation} event below from the noun phrase above:

(6) \text{Positive_regulation}(E1, \text{Trigger} = \textit{mediated}, \text{Theme} = \text{E2}, \text{Cause} = \text{LPS})
\text{Gene_expression}(E2, \text{Trigger} = \textit{expression}, \text{Theme} = \text{TF})

Two verbal triggers (\textit{require} and \textit{involve}) deserve special treatment: in active voice, the preceding argument is the Theme, unlike for other verbs. For example, the prototypical Cause dependency, \textit{nsubj}, indicates a Theme in the following sentence, making the second event (E2) the Theme of the first event (E1):

(7) a. \text{Regulation} of \text{interleukin-1beta transcription} by \text{Epstein-Barr virus} \text{involves} a number of latent proteins via their interaction with \text{RBP}.

b. \textit{nsubj(\text{involves, Regulation})prep_of(Regulation, transcription)amod(transcription, interleukin-1beta)}
c. \text{Regulation}(E1, \text{Trigger} = \textit{involves}, \text{Theme} = \text{E2})
\text{Regulation}(E2, \text{Trigger} = \text{Regulation}, \text{Theme} = \text{E3})
\text{Transcription}(E3, \text{Trigger} = \text{transcription}, \text{Theme} = \textit{interleukin-1beta})

Akin to passive constructions, for these two verbs, we perform simple \textit{argument inversion} in the rules involving Theme and Cause arguments. Some of the argument identification rules are summarized in Figure 2.

Two grammatical constructions are crucial to determining the event participants: coordination and apposition. We summarize how these constructions affect event extraction below.

\(^3\) We use * to refer to incorrect dependency relations or events.
**Figure 2.** Some of the event argument identification rules. Each rule determines the semantic role of Participant for the event triggered by Trigger. ‘*’ is used here in the typical regular expression sense. X represents a token that is neither Trigger nor Participant.
3.4.1. Coordination. Two types of coordination, namely trigger and argument coordination, play a role in event extraction. We rely on dependency relations to determine coordination, considering the participants of a dependency with a type descending from conj (conjunct) (e.g., conj_and, conj_or) to be coordinated. We describe how trigger and argument coordination affect event extraction below:

**Trigger coordination** When the event trigger is conjoined with another word token, dependency relations involving this conjunct are also considered for event argument identification, provided that a biological entity does not intervene between the two conjuncts. This leads us to identify CTCF as an argument of the event triggered by phosphorylation, due to coordination of expression and phosphorylation, in the following example:

   b. prep_of(expression,CTCF)
   c. Phosphorylation(E1, Trigger=phosphorylation, Theme=CTCF)

By disallowing intervention of biological entities, we ensure that a certain class of incorrect event annotations is prevented. For example, in the sentence fragment shown below, where the trigger expression is coordinated with secretion, the event in Example (9b) involving the Theme argument of secretion is prevented, due to this restriction.

(9) a. ...secretion of IL-2 and expression of...
   b. conj_and(secretion, expression)
   c. *Gene_expression(E1, Trigger=expression, Theme=IL-2)

**Entity coordination** When an event is detected and its argument is found to be coordinated with other entities, new events involving the event trigger and each of these entities are also created. An exception are Binding events, which may have multiple Themes. In this case, we simply add conjunct entities as the Themes of the base event. An example of how entity coordination affects event extraction is given below, where the coordination between IL-3 and GM-CSF is considered and event E2 is created due to entity coordination:

(10) a. These findings establish that Rel ... is required by T lymphocytes for production of IL-3 and GM-CSF.
   b. conj_and(IL-3,GM-CSF)
   c. Gene_expression(E1,Trigger=production,Theme=IL-3)
   Gene_expression(E2, Trigger=production, Theme=GM-CSF)

Recognizing that Stanford dependency parsing misses some complex expressions of coordinated entities typical of biological text (in particular, those involving parentheses), we implemented simple heuristics to better resolve coordinated entities. These rules stipulate that entities that have between them:

1. Only a comma (,) or a semi-colon (;)
2. A word with CC (coordinating conjunction) part-of-speech tag
3. A complete parenthetical expression
4. Any combination of the above

are coordinated. For instance, in Example (11), we rely on items (2) and (3) above to recognize the modifier coordination between interleukin-2 and IL-4. Dependency parsing missed this dependency and identified the incorrect dependency relation shown below, instead.

(11) a. The activation of NFAT by TCR signals has been well described for interleukin-2 (IL-2) and IL-4 gene transcription in T cells.
   b. *conj_and(interleukin-2,transcription)

3.4.2. Apposition. Words in an apposition construction are considered interchangeable for event extraction purposes. Therefore, if an entity word is in apposition with a nonentity word that is connected to a trigger with an appropriate dependency path, that entity is marked as the event argument. In the following example, the appos (appositive) and the prep_of dependencies allow us to extract the event E1, as the appos dependency serves to link the event trigger to the head of the protein term:

(12) a. ...upregulation of the lung vascular adhesion molecule, intercellular adhesion molecule-1, was greatly reduced by ...
   b. appos(molecule,molecule-1)
   prep_of(upregulation,molecule)
   c. Positive_regulation(E1, Trigger=upregulation, Theme=intercellular adhesion molecule-1)

The dependencies that we consider to encode apposition constructions are: appos (appositive), abbrev (abbreviation), prep_{including, such as, compared_to, compared_with, versus} (prepositional modifier marked with including, such as, compared to, compared with or versus).

3.5. Postprocessing

After events are extracted using the dependency based grammar and other rules, we apply two postprocessing rules to deal with limitations of our approach and to ensure increased recall.

As mentioned above, we have not considered multiword event triggers. One of the postprocessing rules relaxes this restriction for a limited set of multiword event triggers. If a Regulation event has been identified and the event trigger word is modified by positive or negative (or inflectional forms positively, negatively), the event type is changed to more specific Positive_regulation or Negative_regulation, respectively.

The second postprocessing rule deals with the limitation of not allowing multiple events on the same trigger. Noting that certain trigger words, including overexpression and several others related to transfection (e.g., transfect, transfection, cotransfect) almost always trigger both Gene_expression and Positive_regulation events, we added to the extracted event list a Positive_regulation event if a Gene_expression event was recognized with one of these event triggers, and vice versa.
4. SPECULATION AND NEGATION DETECTION

Once the event annotations are obtained for a sentence, the speculation and negation module determines whether these events are speculated and/or negated. Similar to event extraction, we rely on a small set of speculation/negation cues and dependency-based heuristics that consider the dependencies between the event triggers and speculation/negation cues in the sentence.

One challenging aspect of detecting speculation and negation from the Shared Task perspective is that textual cues indicating these two phenomena are not annotated. However, we have an advantage in this regard, as we worked on the speculation recognition topic prior to the Shared Task and, in fact, developed a dictionary of hedging cues. On the other hand, negation detection is a new topic for us. Considering that, in essence, it is a similar task to speculation recognition, we naturally extended our speculation recognition module to consider negation.

4.1. Speculation Recognition

Our existing speculation recognition module (Kilicoglu and Bergler 2008) relies on a dictionary of weighted hedging cues, induced semi-automatically from linguistic classifications and expanded using lexical resources, and existence of various syntactic patterns to determine whether a sentence is speculative. While its focus was not on identifying speculated events specifically, it lays the groundwork for recognizing speculated events. Consider the example below which illustrates how the existing system naturally extends to identifying events that are speculated:

(13) a. A unique transcriptional mechanism distinct from those regulating the IL-2 or IL-4 genes \textit{seems to control} the IL-5 gene.
    b. \textbf{Regulation}(E1, Trigger=\textit{control}, Theme=IL-5)\newline Speculation(M1, Event=E1)

The existing module already identifies the copular verb \textit{seem} as a hedging cue. Furthermore, the fact that it takes an infinitival complement is taken as a factor strengthening the speculative nature of the sentence. To recognize speculated events, we simply take this one step further and stipulate that this cue/syntactic pattern combination renders speculative the event triggered by the head of the infinitival complement (\textit{control}, in this case).

We refined the existing speculation module in two aspects for the Shared Task. First, we noted that modal verbs (e.g., \textit{may}, \textit{might}) and epistemic adverbs (e.g., \textit{possibly}, \textit{probably}), two classes that we considered strongly speculative, are rarely used to mark speculative contexts in the training corpus. For the Shared Task, we ignore lexical cues in these classes completely for an increased precision score. Secondly, corpus analysis revealed a new syntactic pattern for expressing speculation that we have not considered before. This pattern involves a class of related verbs that we call \textit{active cognition verbs} (e.g., \textit{examine}, \textit{evaluate}, \textit{analyze}, \textit{study}, \textit{investigate}). We search for a Theme dependency pattern between one of these verbs and an event trigger and mark the event as speculated, if such a pattern exists. Nominalizations of these verbs (e.g., \textit{examination}, \textit{evaluation}) are also considered. In Example (13), the \textbf{Regulation} event triggered by \textit{effects} is speculated, because \textit{effects} is the direct object (i.e., a Theme dependency) of \textit{studied}, an active cognition verb. Note that the \textbf{Regulation} event (E2) is recognized due to a “corrective” dependency path (\textit{dobj-prep_on}).
**Table 1.** Negation Cues and the Corresponding Dependencies (xcomp: clausal complement, infmod: infinitival modifier, det: determiner).

<table>
<thead>
<tr>
<th>Negation Cue</th>
<th>Dependency</th>
</tr>
</thead>
<tbody>
<tr>
<td>lack, absence</td>
<td>prep_of(Cue, Trigger)</td>
</tr>
<tr>
<td>unable, &lt;not&gt; able, fail</td>
<td>xcomp(Cue, Trigger)</td>
</tr>
<tr>
<td>inability, failure</td>
<td>infmod(Cue, Trigger)</td>
</tr>
<tr>
<td>no, not, cannot</td>
<td>det(Trigger, Cue)</td>
</tr>
</tbody>
</table>

(14) a. We have studied the effects of prednisone (PDN), ... on the production of IL-2...
   b. dobj(studied, effects)
      *prep_on(studied, production)
      prep_of(production, IL-2)
   c. Gene_expression(E1, Trigger=production, Theme=IL-2)
      Regulation(E2, Trigger=effects, Theme=E1)
      Speculation(M1, Event=E2)

4.2. Negation Detection

Negation detection is similar to speculation recognition. To create a simple negation cue dictionary, we determined several classes of negation cues based on corpus analysis. The scope identification of negation is the same as that of speculation; that is, the negation module negates events if there is an appropriate dependency between one of the negation cues and the event triggers. The lexical negation cues and the dependencies that are sought for them to indicate negation are given in Table 1.

Some typed dependencies implicitly indicate negation. Such dependency types are **neg** (negation) and **conj_negcc** (negated coordination). Participation of an event trigger in dependencies of these types is often sufficient to negate the corresponding event; that is, no specific negation cue is required. For a **neg** dependency to negate an event, the event trigger is required to be one of the participant words of the dependency. On the other hand, for a **conj_negcc** dependency to indicate negation, the event arguments, as well as event triggers, are allowed to be the participants of the dependency. Therefore, in the following example, the event E2 is negated, based on the dependencies in Example (15b). Note that the **conj_negcc** dependency involves event arguments.

(15) a. Exogenous ETS1, but not ELF1, can transactivate GM-CSF, ..., in a PMA/ionomycin dependent manner.
   b. conj_negcc(ETS1, ELF1)
      nsubj(transactivate, ETS1)
      dobj(transactivate, GM-CSF)
   c. Positive_regulation(E1, Trigger=transactivate, Theme: GM-CSF, Cause=ETS1)
      Positive_regulation(E2, Trigger=transactivate, Theme: GM-CSF, Cause=ELF1)
      Negation(M1, Event=E2)
Finally, if none of the above applies and the word preceding the event trigger or one of the event arguments is a basic negation cue (no, not, cannot), the event is negated for an increased recall score.

5. THE PIPELINE

Based on the discussion of the methods in previous sections, our pipeline consists of the following steps, also illustrated in Figure 3:

1. Determine whether a word is an event trigger or not, using the dictionary-based approach.
2. Based on the trigger word and its part-of-speech, determine and apply the dependency-based rules to identify event arguments.
3. Confirm that the event trigger and the potential arguments form a valid event.
   (a) If the event trigger corresponds to a complex regulatory event and it has a potential subevent argument, determine in a recursive fashion whether the subevent is a valid event.
4. Apply postprocessing rules to update events or to identify additional events.
5. Determine whether the sentence contains speculation and negation cues, and if so, determine whether speculation and negation dependency rules hold between these cues and the event triggers.

The first step is a simple dictionary lookup. Provided that a word is tagged as noun, verb or adjective, we check whether it is in our dictionary, and if so, determine the event type it corresponds to. We also ensure that its reliability score is above the given threshold. This word is considered the cue for an event. When the supervised learning scheme is used, the model is applied to the candidate word and the event type, if any, is determined.

We then apply the appropriate dependency-based rules to determine whether any entity or event trigger (in the case of complex regulatory events) in the sentence qualifies as an
argument of the event cue. Grammar rules are applied in the order of simplicity; rules that involve a direct dependency between the cue and any word of the entity are considered first.

Once a list of potential participants is obtained by consecutive application of the rules, one of two things may happen: Provided that subevents are not involved and appropriate arguments have been identified (e.g., a Theme is found for a Localization event), the event is confirmed and is simply added to the extracted event list. Otherwise, we proceed recursively to determine whether the subevent argument can be resolved to an event with protein terms as arguments. If this recursive process yields no such event in the end, the event in question is rejected. In the following example, the event triggered by inhibit is rejected even though its Cause argument, JunB, is recognized, because its Theme, subevent triggered by activation, cannot be assigned a Theme and therefore is considered invalid.

(16) ... , JunB, is shown to inhibit activation mediated by JunD.

After two postprocessing rules are applied, a simple dictionary lookup determines if there are any negation or speculation cues in the sentence. If so, the dependencies involving these cues and the event triggers are assessed for negation and speculation. As mentioned above, negation cues are not required for rules involving neg and conj_negcc dependency relations.

6. EVALUATION

Our event extraction system, with dictionary-based approach to event trigger detection, had one of the best performances in the Shared Task. With the official approximate span matching/approximate recursive matching evaluation criteria, in Task 1, core event detection, we were ranked third, while our speculation and negation detection module performed best among the six systems participating in Task 3. Not surprisingly, our system favors precision, typical of most rule-based systems. Full official results for our submission are presented in Table 2. These results are obtained at the reliability score threshold of .08, found to be optimal. Increasing the threshold increases precision, while lowering recall.

We note that our submission is ranked second overall in Task 1 with the strict matching evaluation criteria. On the other hand, using the event decomposition evaluation criteria, our performance relative to other groups in recognizing Binding events is improved and we place 7th for that event type.

Using the SVM-based classifier to detect event triggers, we obtained the results in Table 3 with the approximate span matching/approximate recursive matching evaluation criteria. The rank column shows the hypothetical position we would get, had we submitted these results. It is easy to see that the supervised model benefits the simple events in terms of recall, while it has a slightly negative effect on recognition of complex regulatory events (particularly on the Regulation event type). However, overall rankings for both event classes are not affected. Interestingly, we rank significantly higher for two event classes that we have not performed well on with the dictionary-based approach (i.e., Binding and Transcription). On the other hand, our performance using the strict matching evaluation criteria is slightly worse than that obtained with the dictionary-based approach (F-score of 0.4169), although overall ranking, again, is not affected.

7. DISCUSSION

In this section, we discuss strengths and limitations of our event extraction system based on evaluation results, providing an analysis of error cases. In addition, we highlight some of our concerns regarding annotation of biological events and speculation.
TABLE 2. Official Evaluation Results, with Approximate Span Matching/Approximate Recursive Matching Evaluation Criteria (EVT-TOTAL: Simple Events and Binding, REG-TOTAL: Complex Regulatory Events, MOD-TOTAL: Speculation and Negation, ALL-TOTAL: All Annotations)

<table>
<thead>
<tr>
<th>Event Type</th>
<th>Recall</th>
<th>Precision</th>
<th>F-score</th>
<th>Rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>Localization</td>
<td>35.63</td>
<td>92.54</td>
<td>51.45</td>
<td>5</td>
</tr>
<tr>
<td>Binding</td>
<td>20.46</td>
<td>40.57</td>
<td>27.20</td>
<td>11</td>
</tr>
<tr>
<td>Gene_expression</td>
<td>55.68</td>
<td>79.45</td>
<td>65.47</td>
<td>5</td>
</tr>
<tr>
<td>Transcription</td>
<td>15.33</td>
<td>60.00</td>
<td>24.42</td>
<td>13</td>
</tr>
<tr>
<td>Protein_catabolism</td>
<td>64.29</td>
<td>56.25</td>
<td>60.00</td>
<td>5</td>
</tr>
<tr>
<td>Phosphorylation</td>
<td>69.63</td>
<td>95.92</td>
<td>80.69</td>
<td>2</td>
</tr>
<tr>
<td>EVT-Total</td>
<td>43.10</td>
<td>73.47</td>
<td>54.33</td>
<td>5</td>
</tr>
<tr>
<td>Regulation</td>
<td>24.05</td>
<td>45.75</td>
<td>31.53</td>
<td>1</td>
</tr>
<tr>
<td>Positive_regulation</td>
<td>28.79</td>
<td>50.45</td>
<td>36.66</td>
<td>2</td>
</tr>
<tr>
<td>Negative_regulation</td>
<td>26.65</td>
<td>51.53</td>
<td>35.13</td>
<td>3</td>
</tr>
<tr>
<td>REG-Total</td>
<td>27.47</td>
<td>49.89</td>
<td>35.43</td>
<td>2</td>
</tr>
<tr>
<td>Negation</td>
<td>14.98</td>
<td>50.75</td>
<td>23.13</td>
<td>1</td>
</tr>
<tr>
<td>Speculation</td>
<td>16.83</td>
<td>50.72</td>
<td>25.27</td>
<td>1</td>
</tr>
<tr>
<td>MOD-Total</td>
<td>15.86</td>
<td>50.74</td>
<td>24.17</td>
<td>1</td>
</tr>
<tr>
<td>ALL-Total</td>
<td>32.68</td>
<td>60.83</td>
<td>42.52</td>
<td>3</td>
</tr>
</tbody>
</table>

TABLE 3. Evaluation Results with the Supervised Event Trigger Detection Model

<table>
<thead>
<tr>
<th>Event Class</th>
<th>Recall</th>
<th>Precision</th>
<th>F-score</th>
<th>Rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>Localization</td>
<td>33.91</td>
<td>88.06</td>
<td>48.96</td>
<td>5</td>
</tr>
<tr>
<td>Binding</td>
<td>23.34</td>
<td>41.75</td>
<td>29.94</td>
<td>8</td>
</tr>
<tr>
<td>Gene_expression</td>
<td>55.96</td>
<td>80.00</td>
<td>65.85</td>
<td>5</td>
</tr>
<tr>
<td>Transcription</td>
<td>36.50</td>
<td>46.30</td>
<td>40.82</td>
<td>5</td>
</tr>
<tr>
<td>Protein_catabolism</td>
<td>64.29</td>
<td>60.00</td>
<td>62.07</td>
<td>4</td>
</tr>
<tr>
<td>Phosphorylation</td>
<td>70.37</td>
<td>95.00</td>
<td>80.85</td>
<td>2</td>
</tr>
<tr>
<td>EVT-Total</td>
<td>45.65</td>
<td>70.58</td>
<td>55.44</td>
<td>5</td>
</tr>
<tr>
<td>Regulation</td>
<td>21.99</td>
<td>42.38</td>
<td>28.96</td>
<td>4</td>
</tr>
<tr>
<td>Positive_regulation</td>
<td>28.28</td>
<td>50.27</td>
<td>36.20</td>
<td>2</td>
</tr>
<tr>
<td>Negative_regulation</td>
<td>28.50</td>
<td>48.43</td>
<td>35.88</td>
<td>2</td>
</tr>
<tr>
<td>REG-Total</td>
<td>27.22</td>
<td>48.54</td>
<td>34.88</td>
<td>2</td>
</tr>
<tr>
<td>Negation</td>
<td>14.54</td>
<td>47.83</td>
<td>22.30</td>
<td>1</td>
</tr>
<tr>
<td>Speculation</td>
<td>17.79</td>
<td>50.00</td>
<td>26.24</td>
<td>1</td>
</tr>
<tr>
<td>MOD-Total</td>
<td>16.09</td>
<td>48.95</td>
<td>24.22</td>
<td>1</td>
</tr>
<tr>
<td>ALL-Total</td>
<td>33.67</td>
<td>59.15</td>
<td>42.92</td>
<td>3</td>
</tr>
</tbody>
</table>

7.1. Event Triggers

7.1.1. Dictionary versus Supervised Model. Two distinct approaches to event trigger detection lead to interesting results. The dictionary-based approach, based on several principles and careful generalizations as well as on simple statistics, yields results comparable to
those obtained with a standard machine learning approach. The supervised model for event triggers is devised to address limitations of the dictionary-based approach, mainly recall. While we observe a slight increase in overall recall with the supervised model, this effect is mostly limited to simple events. During the development of our system, we placed greater emphasis on the structural adequacy of our treatment of complex regulatory events. The usefulness of the supervised model in identifying simple event triggers, but not so much in complex regulatory event triggers, shows that a supervised model finds omitted patterns in appreciable quantity for simple events but cannot compensate for the sophisticated linguistic generalizations of a human knowledge engineer when it comes to complex events.

It can be argued that a more predictive supervised model could be built using a larger, more optimal feature set, leading to a better event extraction performance. While this may be true, we emphasize that we do not aim for such feature engineering. Our goal in this experiment was to simply assess whether we can overcome the limitations of the dictionary-based approach, with a significant improvement in event extraction performance that would justify the complexity and the cost of training a supervised model. In this regard, it seems fair to say that the improvement we obtained in overall performance is not sufficient. The process of inducing the event trigger dictionary is semi-automatic and takes a very short time and minimal computational effort, leading us to conclude that the dictionary-based approach provides higher return on investment.

7.1.2. Errors due to Event Trigger Detection. In the dictionary-based approach, there are several sources for errors. We miss certain event triggers simply because of our simplifying assumptions, discussed in Section 3.2. We do not discuss such errors here.

The simplistic view of event triggers (“once a trigger, always a trigger”) leads to false positive errors. One such example is given in Example (17):

(17) a. We show that... LPS treatment enhances the oligomerization of TLR2.
    b. Binding(E1, Trigger=oligomerization, Theme=TLR2)

where we identify the event shown in Example (17b) from the sentence fragment in Example (17a). We consider oligomerization a reliable trigger for Binding, because it occurs twice in the training corpus, both times as event triggers for this event type. However, in this instance, it does not trigger an event. We could limit event triggers to those occurring more than a specified number of times; however, that would result in even lower recall.

The dictionary-based approach is naturally limited and will lead to recall errors, in which we do not recognize an event trigger as such, simply because we have not encountered it in the training corpus, or it does not have an appropriate part-of-speech tag. However, the intuitive, explicit nature of the dictionary-based approach can overcome this limitation through generalizations and observations outside the training data by a domain expert, leading to a system that will adapt and improve over time.

7.2. Argument Identification

7.2.1. Dependency Relations. Our results confirm the usefulness of dependency relations as foundation for event extraction systems. By considering a total of 33 dependency path based rules (6 of which are “corrective,” compensating for parser errors), we are able to obtain very competitive results. There is clearly much room for improvement, particularly in terms of recall, and we believe that the small number of rules involved and incremental nature of rule development accommodates such improvements fairly easily.
One of our main motivations for participating in the Shared Task was to assess our existing speculation recognition module in Task 3. Recognizing that most of the speculation instances in the training corpus involved complex regulatory events, we chose to concentrate on this class of events, rather than on simple events. Rules involving “corrective” dependency paths were implemented as part of this special focus. We find that they are useful in overcoming the weaknesses of the Stanford dependency parsing, contributing to our near state-of-the-art system performance in identifying arguments of complex regulatory events.

Although we did not focus specifically on simple events, dependency relation based heuristics seem sufficient for identifying their Theme arguments. One simple event type we have not performed well on is Transcription; however, the improvement we obtain in this event type when the supervised model is used for trigger detection suggests that the poor performance is mainly due to the weakness of the dictionary-based approach in detecting triggers of this event type. Furthermore, the unusually large gap between our performance on the development data and test data concerning this event type (F-score of 0.5821 and 0.2442, respectively) indicates that the trigger expressions derived from the training data for this event type are largely predictive of the triggers in the development corpus, but not so of those in the test corpus.

The Binding event type exhibits additional complexity, because it can take multiple Theme arguments. We performed relatively poorly in this event type. This seems to be partly due to our lack of effort to distinguish Binding event instances by the number of arguments. Our relatively improved Binding performance in the event decomposition mode and the good results reported with participating systems that differentiate Binding events by the number of participants involved (Van Landeghem et al. 2009; Sætre et al. 2009) seem to support this finding.

7.2.2. Errors due to Dependency Relations. Accurate dependency parsing of biological text is challenging, due to the complexity of such text. Our system generates some false positive errors due to dependency parsing errors. For instance, in Example (18a), there is a noun phrase coordination between the phrases NF-kappaB nuclear translocation and transcription of E-selectin and IL-8. However, this construction is missed and an erroneous prep_of dependency is generated, attaching E-selectin to translocation rather than to transcription, leading to two false positive errors shown in Example (18c).

(18) a. ...leading to NF-kappaB nuclear translocation and transcription of E-selectin and IL-8, which results in ...

b. conj_and(transcription, translocation)
   prep_of(translocation, E-selectin)
   conj_and(E-selectin, IL-8)

c. *Localization(E1, Trigger=translocation, Theme=E-selectin)
   *Localization(E2, Trigger=translocation, Theme=IL-8).

More “corrective” dependency paths can be employed to correct such errors; however, first, a closer examination of such error patterns is necessary.

In other instances, the required dependency is completely missed by the parser, leading to false negative errors. For instance, in Example (19a), we are unable to recognize two events shown in Example (19c) due to the lack of apposition-indicating dependency relations between repressors and 4E-BP1 or 4E-BP2.
(19) a. ... specific regulation of two repressors of translation initiation, 4E-BP1 and 4E-BP2.
   b. prep_of(regulation, repressors)
      prep_of(repressors, initiation)
      *conj_and(intiation, 4E-BP1)
      *conj_and(initiation, 4E-BP2)
   c. Regulation(E1, Trigger=regulation, Theme=4E-BP1)
      Regulation(E2, Trigger=regulation, Theme=4E-BP2)

Typical of rule-based systems, we miss events expressed using rare patterns. Event arguments expressed as heads of prepositional phrases marked with from is one such case. An example is given below:

(20) Calcineurin activates transcription from the GM-CSF promoter ...

In this case, the event Transcription(Trigger=transcription, Theme=GM-CSF) is missed. It is fairly easy to add a rule to address such occurrences; however, more examples need to be analyzed first. Subcategorization frames of trigger words could also be gainfully employed for a more general solution to such cases.

7.3. Speculation and Negation

Our system performed best in Task 3 with a large margin. We attribute this to the portability and extensibility of our prior work in speculation recognition and usefulness of dependency relations in scope identification. Our special focus on complex regulatory events also contributes to our performance, as 70% of speculation instances involve such events.

Although our system performed best, evaluation results regarding Task 3 shown in Table 2 and Table 3 may still seem very poor (F-score of 0.2417 and 0.2422, respectively). However, we find that most of the errors concern misidentified or missed base events. To specifically assess our performance on speculation and negation recognition, we further analyzed our results on the development corpus. Considering only the speculation and negation instances associated with correctly identified base events, we obtain the results shown in Table 4.

On the test corpus, somewhat lower accuracy figures are to be expected. However, official evaluation results confirm that our approach to speculation and negation detection is robust and state-of-the-art.

7.3.1. Speculation Recognition Errors. Analysis of speculation errors on the development data set shows that false positive errors are exclusively cases for which speculation
could be argued. For instance, in Example (21), we recognize that the speculation cue *appears* scopes over the Negative_regulation event in Example (21c), making it speculative. However, speculation is not annotated in this instance.

(21) a. An unidentified Ets family protein binds to the EBS overlapping the consensus GAS motif and *appears* to negatively *regulate* the human IL-2R alpha promoter.
   b. xcomp(appears,regulate)
      dobj(regulate,promoter)
      nn(promoter,alpha)
   c. Negative_regulation(E1, Trigger=negatively_regulate, Theme=IL-2R alpha)?
      Speculation(M1, Event=E1)

Complex and rare patterns of expressing speculation are responsible for speculation recall errors, an example of which is given below:

(22) a. These results...suggest a molecular mechanism for the inhibition of TLR2 by DN variants.
   b. dobj(suggest,mechanism)
      prep_for(mechanism,inhibition)
   c. Negative_regulation(E1, Trigger=inhibition, Theme=TLR2)
      Speculation (M1, Event=E1)

In this example, *suggest* is detected as a speculation cue; however, we are unable to recognize that it scopes over the event triggered by *inhibition*, because we do not consider the dependency path involving *dobj* and *prep_for* to identify speculated events.

7.3.2. Negation Detection Errors. Examining the negation errors on the development data set, we find that the negation pattern that involves simple negation cues (*no, not, cannot*) preceding an event trigger or an argument, a pattern initially considered to increase recall, causes most of negation false positive errors. An example is presented below:

(23) a. The finding that HL-60/vinc/R cells respond to TPA with induction of a monocytic phenotype, but not c-jun expression, suggests that...
   b. Gene_expression(E1, Trigger=expression, Theme=c-jun)
      Positive_regulation(E2, Trigger=induction, Theme=E1)
      *Negation (M1, Event=E1)

Here, it is not the factuality of the event E1 that is reversed with negation, but that of E2. Similar to speculation recall errors, complex and rare patterns of negation expressions, one of which is given below, are responsible for most negation recall errors.

(24) a. Galectin-3 is...and is expressed in many leukocytes, with the notable exception of B and T lymphocytes.
   b. Gene_expression(E1, Trigger=expressed, Theme=Galectin)
      Negation(M1, Event=E1)

In this example, the prepositional phrase, *with the notable exception*, is a complex negation cue that we have not considered.
7.4. Annotation Issues

Event annotations in the Shared Task data are based on the annotation scheme adopted by the GENIA event corpus (Kim et al. 2008), with small differences regarding biological entity annotations. We based our system on careful analysis of the training and development data sets, which revealed some interesting aspects of event annotations, particularly regarding triggers.

As was briefly mentioned in Section 3.2, we find some of the event trigger annotations tenuous; in other words, they are based more on the context surrounding the trigger or on annotator’s inference than on any linguistic notion of event expression. In Example (25), *by* is annotated as trigger for a Positive_regulation event; however, it seems that the entire prepositional phrase introduced with *by* as well as *induction* contribute to trigger such an event.

(25) a. These data suggest a general role for Tax induction of *IL-1alpha* gene transcription by the NF-kappaB pathway.

b. Positive_regulation(E1, Trigger=by, Theme=E2)
   Transcription(E2, Trigger=transcription, Theme=IL-1alpha)

On the other hand, annotator’s inference plays a role in annotating *following* as trigger for a Positive_regulation event in the following example, where temporal ordering of events seems to be taken as causation.

(26) a. Differential induction of interferon (IFN)-inducible protein 10 following differentiation of a monocyte...

b. Positive_regulation(E1,Trigger=induction,Theme=interferon(IFN)-inducible protein 10)
   Positive_regulation(E2,Trigger=following,Theme=E1)

In fact, Kim et al. (2008), in motivating the GENIA event annotation scheme, acknowledge this phenomenon, stating that the event annotations are information-centered, without explicit reference to syntactic or frame-semantic theory. We argue that while this approach may be considered appropriate for practical biological tasks, it ignores the insights that can be gained from a more linguistically informed approach to events. We particularly find the event definition of TimeML (Pustejovsky et al. 2005), the specification language for events and temporal expressions, and its annotation guidelines sensible and beneficial and think that the principles laid out by the TimeML specification would be informative for annotating biological events. This is not to question the value of the GENIA event annotation effort, but to stress the value that linguistic awareness could add to such efforts in the biomedical domain.

Annotation of speculation is another contentious issue. We briefly mentioned in Section 4.1 that we did not consider two classes of speculation cues (*modal verbs* and *epistemic adverbs*) we identified in prior work, because they were rarely annotated to indicate speculation in the Shared Task data. This is in contrast to the Hedge Classification data set (Medlock and Briscoe 2007), the corpus that we used in prior work. Another data set in which speculation is annotated is the BioScope corpus (Vincze et al. 2008). Speculation annotation in this corpus is closer to that in the Hedge Classification data set, while there are interesting, and somewhat curious, differences, such as speculation annotations that involve conjunctions (i.e., *or, either, versus*). On the other hand, the GENIA event annotation
takes constructions involving *active cognition verbs* as indicating speculation, while they are not annotated as such consistently in the other two corpora. These fundamental differences between the corpora and basic definitions demonstrate the lack of a standardized notion of speculation. However, the problem is hardly limited to the biomedical domain. Linguistic classifications addressing speculation and related phenomena (e.g., *evidentiality, modality*) are often ambiguous and fuzzy, providing further evidence for difficulty of annotating such phenomena correctly and consistently. Annotation of speculation (or negation) *scope* is yet another problematic area. In the GENIA event annotation, scope is taken to be an abstract semantic unit (an event), while the BioScope corpus defines it as a textual span. Our view of scope is similar to that adopted by the GENIA event annotation, a view consistent with the TimeML specification for annotating event modality. These basic disagreements among the biomedical corpora highlight the need for the community-wide development of consistent definitions and annotation guidelines for these phenomena.

8. CONCLUSIONS AND FUTURE WORK

We explored a rule-based approach that relies on a dictionary of event triggers and a grammar based on typed syntactic dependency relations and evaluated our approach in the BioNLP’09 Shared Task on Event Extraction. This study marks our first foray into bio-event extraction in a general way and, thus, we consider the results very encouraging. In one area we investigated before, speculation detection, our system performed best, confirming the portability of our approach.

Modest recall figures obtained in the shared task point to areas of improvement. One of these areas is event trigger detection, and we explored supervised machine learning for this subtask. The results we obtained are comparable to those obtained using a dictionary of triggers; however, the computational cost and complexity seem to favor a simple dictionary-based trigger detection.

Our careful analysis of the Shared Task data reveals some issues in annotating bio-events and suggests that the bioNLP community can benefit from the efforts of the larger linguistics community (such as TimeML) in resolving these issues and developing consistent definitions and guidelines.

We did not attempt to address several notions affecting event extraction. First, events spanning multiple sentences (approximately 5% of all events in the Shared Task data) were ignored to reduce the complexity of the problem. Secondly, we did not resolve anaphoric expressions, which led to a fair number of recall errors. We expect that several studies addressing anaphora resolution in biomedical text (Castaño, Zhang, and Pustejovsky 2002; Gasperin and Briscoe 2008) may inform our near future efforts in this area. Finally, dependency relations based on the Stanford Parser provided better performance in our case, in contrast to general consensus that those based on the Charniak Parser (Charniak and Johnson 2005) are superior, and we plan to investigate this curious fact in the future.

REFERENCES


