國立臺灣師範大學
資訊工程研究所碩士論文

指導教授：Wen-Juan Hou 博士

Event Extraction for Gene Regulation Network Using Statistical and Semantic Approaches

研究生： Bamfa Ceesay 撰

中華民國 一0三 年 六 月

ABSTRACT

Genic regulation networks are the primary study object in systems biology. They allow better understanding of the relationship between molecular mechanisms and cellular behavior. However, one of the bottlenecks in systems biology is the acquisition of an accurate genetic regulation network. In the recent years, the BioNLP community has produced systems for extracting genic interactions and Protein-Protein Interaction (PPI) from the literature. The sporulation network of the bacteria model for bacillus subtilis is very well studied. The automatic design of the gene regulation network is one of the main challenges in biology, because it is a crucial step forward in understanding the cellular regulation system.

In this study, we present a description of a system on Gene Regulation Network (GRN) in bacteria and we use the data from the BioNLP'13 shared task (BIONLP-ST) on Event Extraction. For this work, we first propose a procedure to do biological event extraction combining a dependency graph-based method and a method using semantic analysis in Natural Language Processing (NLP). Then a second design, a statistical approach using Hidden Markov Model (HMM), is experimented.

Dependency parsing is a significant and commonly used approach to finding out the dependency relationship between tokens in, for example, a sentence. We use dependency features to identify and classify our event trigger tokens using multi–class Support Vector Machine (SVMLight multiclass). However, the dependency features are not sufficient to give the semantic relationship between tokens with a sentence. Therefore, we develop a semantic analysis approach based on NLP techniques to capture more detail information and improve our result on event extraction.

In our second design approach, we use a general statistical method via Markov's logic instead of developing certain inferences and learning algorithms. Markov's Model has achieved significant recognition in Natural Language Processing especially in the field of speech recognition.

Our result shows that the graph-based approach obtains a better result on event extraction and produces a much better regulation network than the semantic analysis method. The combination of the two approaches has yet a much slightly better result than that with the individual approach. Moreover, the proposed statistical approach achieves a much better result than the combined and individual results of our graph-based and semantic analysis approaches.

Keywords: Biological event extraction, Gene regulation network, Graph-based approach, Semantic approach, Statistical approach

Acknowledgments

Foremost, I would like to express my sincere gratitude to my advisor prof. Wen Juan Hou for the continuous support of my master study and research, for her patience, motivation, enthusiasm, and emmense knowledge. Her guidance helped me in all the time of the research and writing of this thesis. I could not have imagined having a better advisor and mentor for my master study.

I thank my fellow labmates in the Natural Language Processing lab: Tsung Cheng Shang, and Bo Yuan Guo for their stimulating discussion, hours of working together to achieve our academic goals.

Last but not the least; I would like to thanks my parents: Sainey Ceesay and Nghansa Fatty for given birth to and supporting me.

Table of Contents

List of Tables

List of Figures

1 Introduction

Genic regulation networks are the primary study object in system biology. They allow better understanding of the relationship between molecular mechanisms and cellular behavior. However, one of the bottlenecks in systems biology is the acquisition of an accurate genec regulation network. In the recent years, the BioNLP community has produced systems for extracting genetic interactions and Protein-Protein Interaction (PPI) from the literature (Martinez and Baldwin, 2011). The sporulation network of the bacteria model for bacillus subtilis is very well studied. The automatic design of gene regulation network is one of the main challenges in biology, because it is a crucial step forward in understanding the cellular regulation system.

The BioNLP Shared Task (BioNLP-ST) series represents a community-wide trend in text mining for biology toward the fine-grained information extraction (IE)[1]. The BioNLP-ST 2013 follows the general outline and goals of the previous task. The task broadens the scope of the text mining application domains in biology by introducing new issues on cancer genetics and pathway curation. BioNLP-ST 2013 features six event extraction tasks, which are as follows: [GE] Genia Event Extraction for NFKB knowledge base construction, [CG] Cancer Genetics, [PC] Pathway Curation, [GRO] Corpus Annotation with Gene Regulation Ontology, [GRN] Gene Regulation Network in Bacteria, and [BB] Bacteria Biotopes (semantic annotation by ontology). The study focuses on the task of genetic regulation network in bacteria (GRN).

The purpose of GRN task is to retrieve all the genetic interactions of the reference networks; at least one occurrence per interaction independently of where they are

[1] http://2013.bionlp-st.org/

mentioned in the literature; and to present these results as a gene regulation network, and to evaluate the IE systems by their capacity to reconstruct the network.

In systems biology, the term entities are used to refer to the biomolecules and genic objects; and events refer to the low-level molecular and genic events. Data in the GRN task consists of selected sentences from PubMed abstracts. The GRN task is a relation extraction task that follows the BioNLP-ST 2013 frame of representation. The participants are also provided with a manually curated annotation of the training corpus including events, entities, coreferences and genic interactions. For training, participants get a regulation network built from the given sentences in the training data.

The proposed systems depend heavily on the machine learning techniques and an array of features derived from analysis of each given sentence. Our approach is based on three design decision: (1) event extraction using graph-based feature sets, (2) semantic annotation based on natural language processing, and (3) statistical event extraction based on Markove's Logic (Richardson and Domingos, 2006; Riedel *et al*., 2011). Our approach has yielded two separate systems; the first system is a pipeline of subsystems: event extraction using graph-based feature sets and semantic annotation based on natural language processing. Both subsystems are a pipeline of several major processing steps. We test the result of extracted events and their arguments of our individual components. The results are combined to improve the recall and reduce the slot error in the gene regulation network. The results from the two approaches (event extraction using graph-based feature sets and semantic annotation based on natural language processing) are compared and their independent contribution to our final network is determined.

Our third design choice is the following. Instead of designing and implementing the specific inference and training methods for structured models, the proposed study uses Markov Logic, a statistical relational learning method, and defines our model declaratively. Finally, the study represents event structures as the relational structures over tokens of a sentence.

The rest of the thesis is organized as follows. Section 2 gives the introduction of the related works. Section 3 presents the overview of our system architecture and the experimental data. We describe the proposed methods in details and the results achieved by the proposed methods shown with discussion in Sections 4, 5 and 6. Finally, the main conclusion is presented and the future work is outlined.

2 Related Works

There are several pioneering works in the development of event extraction systems for biological literatures. The related works are introduced as follows.

Björne *et al*. (2011) proposed an approach to extract the complex biological events using the syntactic dependency graph. They made extensive use of the dependency analysis of sentences to derive some features. They enhanced their event extraction by using semantic and syntactic features of tokens. They finally used multiclass Vector Support Machine (Tsochantaridis *et al*., 2004) to classify and extract events.

In similar spirit, Huang *et al*. (2006) suggested an automatic extraction of information about molecular interactions in biological pathways from texts based on ontology and semantic processing. They developed a framework using the ontology inference and semantic processing techniques to extract biological knowledge from a large scale of literature in NCBI PubMed.[2] The system integrated various thesauri of WordNet[3], MesH[4] and GO (Gene Ontology)[5].

Markov's Logic has been significantly applied in the Natural Language Processing domain, especially in the speech recognition. Riedel *et al*. (2011) described a biological event extraction system that used Markov's logical networks. Instead of using inferences and learning algorithms, they used Markov's Logic to define statistical relations. They also designed a joint probabilistic model over events in the sentence and represented events as the relational structures over the tokens of a sentence.

[2] http://www.ncbi.nlm.nih.gov/pubmed
[3] http://wordnet.princeton.edu/
[4] http://www.ncbi.nlm.nih.gov/mesh
[5] http://www.geneontology.org/

3 Architecture Overview

3.1 Overall Architecture

Figure 1 shows the overall architecture of our method for the biological event extraction and gene regulation network. For a given sentence, the first processing step is parsing it using the Stanford full parser (McClosky *et al*., 2011)[6]. In the event extraction stage using graph-based feature sets as shown in Part A of Figure 1, the subsystem makes use of some key statistical dependency features such as the shortest path from a token node, bigrams, and trigrams to detect and classify our event triggers and edges (event arguments). In the semantic processing phase, the study analyzes the semantic relationships between the detected triggers and event triggers. Event triggers with improper arguments or with no arguments will be pruned.

In our semantic annotation based on natural language processing method, we focus on using syntactic and semantic features to extract events and event arguments from the parsed sentences (Bui and Sloot, 2011). In the result of extracted events and event arguments, we combine the results from Part A and Part B of Figure 1 via the stacking and filtering strategies.

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

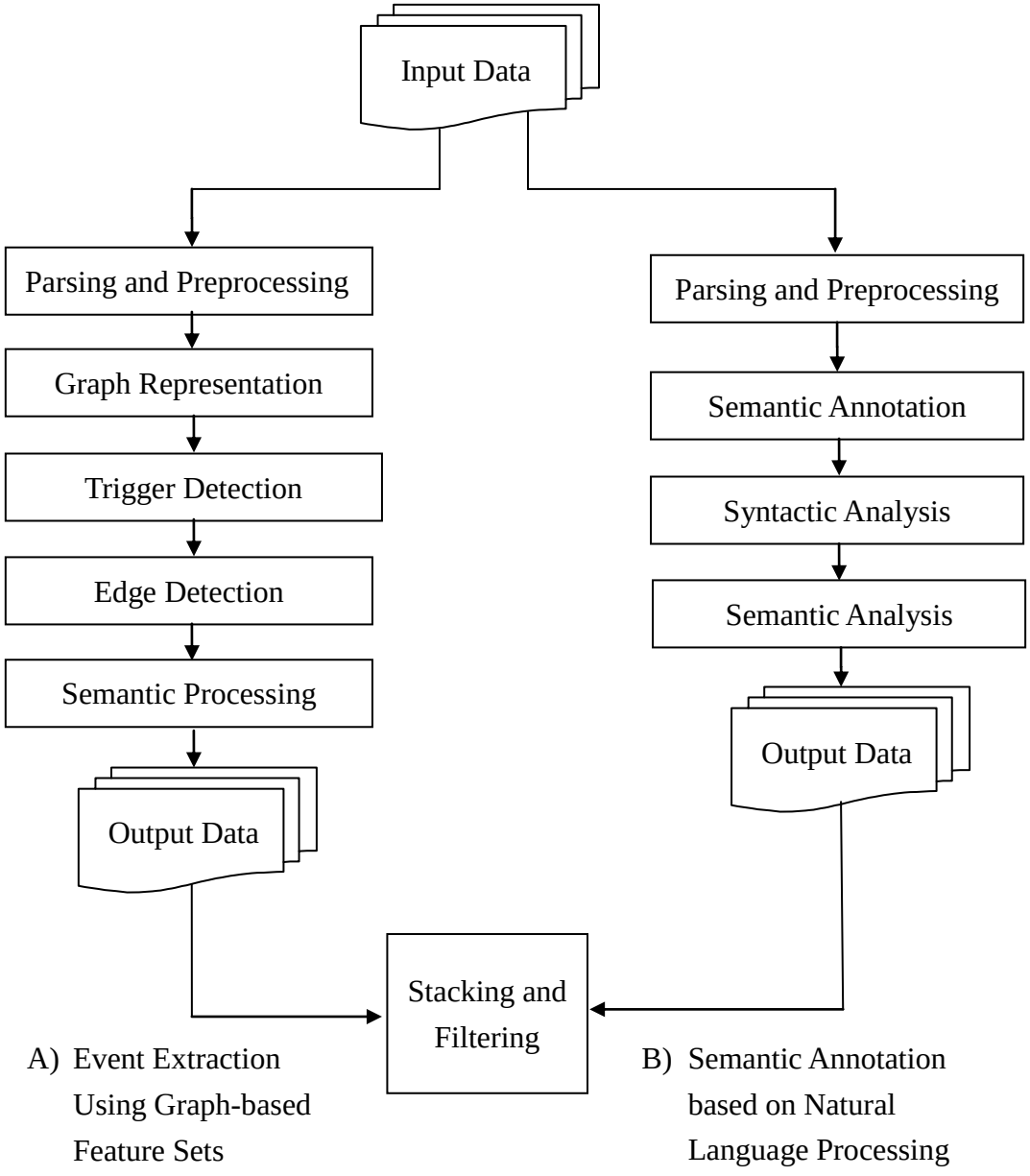

Figure 1. The main components of our system

3.2 Experimental Data

The experimental data are the data given in the BioNLP 2013 shared task. The experimental data consist of three parts: training data, development data, and the test data. The experimental data use the standoff format, separating the text of the documents from annotations. The annotations are connected to specific spans of texts through character offset, and each annotation is associated to their corresponding text file using file naming convention based on the base name (file name without suffix). The base text file and its associated annotation files have the same base name: for example, the file “PMID-1000.a1” contains annotations for the file “PMID-1000.txt.” The file format in the experimental data is identified by the suffix in the filename. The suffix “.txt”, “.a1” and “.a2” is used to represent text files, entity annotation files and event annotation files respectively. The text file consists of only one sentence. Each text file is associated to an annotation file. There are two different annotations in the shared task, the event annotation and the entity annotation. The text files in the training and development data have both associated annotation files. However, the text files in the test data only have their associated entity files. Therefore, the task in the shared task is simplified to finding the event annotations for the test data.

4 Event Extraction Using Graph-Based Feature Sets

This component is a pipeline of three main processing steps: trigger annotation, edge detection and semantic processing. We represent the extraction targets in terms of semantic graphs. The nodes of the graph are an instance of a named entity or a biological event; the edges of the graph represent possible relations that might exist between any given nodes. This summarizes the GRN (Gene Regulation Network in Bacteria) tasks problem to finding nodes and edges of the semantic graph representation for a given input data. The illustration is shown in Figure 2 below.

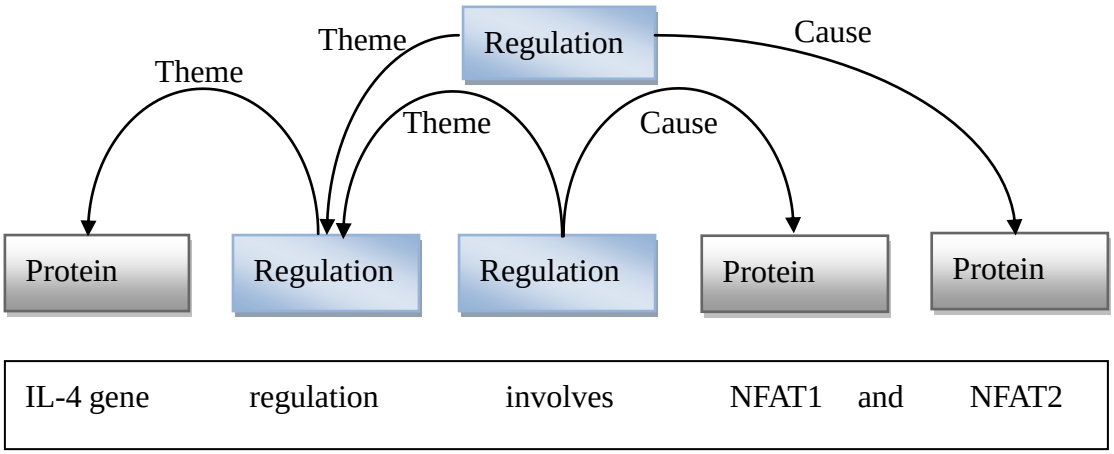

Figure 2. Semantic graph produced by trigger and edge detection

In Figure 2, rectangles filled with grey colors represent named entities. Rectangles filled with blue colors represent events. “Theme” and “Cause” represent the task specific arguments or roles respectively. For a given literature “IL-4 gene regulation involves NFAT1 and NFAT2” as shown in Figure 2, each token is tagged as an event class or a named entity. Tokens of the same event class are represented by one case. In Figure 2, “IL-4 gene”, “NFAT1” and “NFAT2” represent the named entities of type “Protein”. The clue words “regulation” and “involves” describe a “Regulation”

event. Each edge is classified as theme, cause or negative if there is no edge. From Figure 2, it can be express that “IL-4 gene” is the argument of a regulation event and as the object of the expression, has a role (theme) regulation. On the other hands, “NFAT1” and “NFAT2” also are arguments of a regulation event but as the subjects of the expression. Hence, in simple terms, Figure 2 can be express as:

1. *<regulation, theme: IL_4 gene>, regulation of IL-4 gene*
2. *<regulation, cause: NFAT1>, regulation required NFAT1*
3. *<regulation, cause: NFAT2>, regulation required NFAT2*

Moreover, an event can take another event as an argument, and a node can represent multiple events of the same class. As shown in Figure 2, the regulation event is nested to another event of the same class. In such a case, we will use a heuristic rule for the node duplication during the semantic processing phase that will be explained in Section 4.5.

4.1 Parsing and Preprocessing

Our preprocessing includes splitting the input sentences into a set of terms, replacing all identified named entities with placeholders and parsing the sentences with Stanford dependency parser. First, we replace all the named entities with their given annotated IDs in the given gold standard entity annotation. This helps prevent our parser from segmenting multiple word protein named entities. Finally, the input sentence is parsed to produce a dependency parse tree.

In an attempt to improve trigger detection as discussed in Section 4.3, we create a dictionary of event triggers from the gold standard annotations in the training and

development data. For tokens in our input sentence, we detect candidate triggers from our dictionary and determine its trigger type. This step helps us obtain a list of candidate triggers for a given input and their probability score for each event type. This score is calculated as the probability of a candidate trigger, given a trigger type.

Figure 3 shows an example of parsing and preprocessing a sentence "IL-4 gene regulation involves NFAT1 and NFAT2."

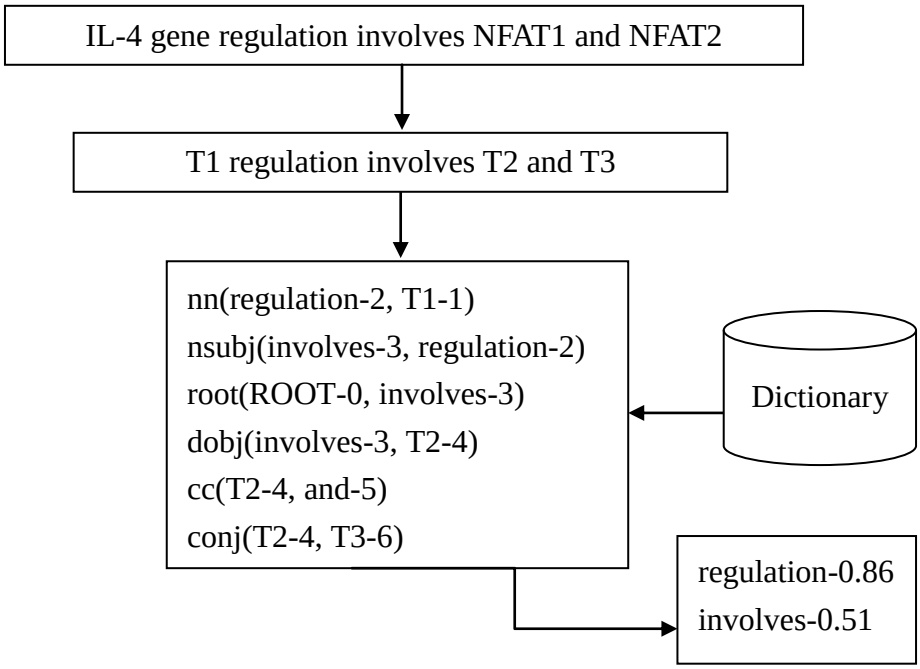

Figure 3. An example of a parsed and preprocessed sentence

In the above illustration, T1, T2 and T3 are placeholders, specifically identifying the entity IDs in the entity annotation file. As a result, from the preprocessing, we identify two event triggers, "regulation" and "involves" with weights, *w*, of 0.86 and 0.51, respectively. In Figure 3, the weight *w* is calculated as below.

$$w = (\text{fraction of event token in sentence tokens}) + (\text{fraction of event token in background data tokens}) \quad \textbf{(1)}$$

From our parsed tree in Figure 3, the expressions are explained as follows. “nn(regulation-2, T1)” is a noun compound modifier relation between regulation and token T1. “nsubj(involves-3, regulation-2)” is a nominal subject relation between a nominal subject (regulation) and the governor verb (involves). “root(ROOt-0, involves-3)” is the root word of our parsed tree. “dobj(involves-3, T2-4)” represents a dependency relation of direct object (dobj) T2 of the verb (involves). “cc(T2-4, and-5)” is a coordination relation showing the relationship between the element of conjunct (T2) and the coordinating conjunction (and). And “conj(T2-4, T3-6)” is a conjunction relation between two terms (T2 and T3) connected by a coordinating conjunction. The number in the parse tree result represents the position of the corresponding token in the sentence. For example, T2-4 implies that token T2 is at position 4 of the sentence.

4.2 Graph Representation

In the graph representation, we represent the nodes in the graph as named entities or events, and the edges of the graph represent the event arguments. This helps us simplify the task to finding the nodes and edges of the graph to find the events and event arguments. Graph representation of an example sentence is illustrated in Figure 2.

4.3 Trigger Detection

A trigger is any word in a sentence that serves as an indicator to an event. Due to the nature of event annotation of the biological literature, the trigger detection cannot be simply reduced to a dictionary lookup for trigger expression for certain reasons. First, a number of common textual expressions act as event triggers in some cases, but fail to

act as triggers in other cases. Second, a single expression may be associated with various event classes. For example, the instances of the token "*over expression*", are evenly distributed among "*positive regulation*," "*transcription*" and the negative classes. In light of these challenges, we approach trigger detection with a multi-class support vector machine (SVMLight multiclass) classifier that assigns event classes to individual tokens, one at a given time instance.[7] This is in contrast to the sequence-modeling problem, which applies a sequential model.

The study implements the trigger detection as a token labeling problem, by assigning an event class to tokens or a negative class if it cannot be assigned to any event class. This implies that for a given token, if it is assigned to a class, it is therefore a trigger; else, it is a non-trigger. In some instances, adjacent tokens belong to the same class. We will assign a single trigger to such adjacent tokens if and only if one of the token occurs in the training data. It is observe that different event classes could share the same token as a trigger. Therefore, such token belongs to more than one event classes. Hence, the study uses the multi-class classification approach for our trigger detection. We use the combined class to assign detected triggers to a single event class (McClosky *et al.*, 2012). For instance, "*transcription/regulation*" denotes a token that acts as a trigger to two events of the mentioned classes. This implies that, our trigger detection produces a single event node for each detected trigger.

The classifier is trained on gold standard triggers from the training data and incorporates several arrays of the token to be classified. Table 1 below shows features used in the trigger detector.

[7] http://www.cs.cornell.edu/people/tj/svm_light/svm_multiclass.html

Table 1. Features of trigger candidate for trigger detection

Type	Features
Token	Token features are listed in Table 2 An example is illustrated in Table 3
Candidate in the parser output	Token features of word with dependencies from candidate Bigrams of dependencies Bigram and Trigram of words (base form + POS) An example is illustrated in Table 4
Shortest path	Shortest path features listed in Table 5 between candidate and their closest entity Length of the path between candidates and entities An example is illustrated in Table 6

Token features include features of the individual token. They are testing for token capitalization, testing for punctuation and numeric characters, testing for stemmed tokens and token frequency. In trigger detection, the state (i.e., presence or absence) of the token in known trigger expressions and their classes extracted from the training corpus are used. In general, Table 2 shows our token features. Table 3 shows the result of token features for "IL-4 gene regulation involves NFAT1 and NFAT2".

Table 2. List of token features

- Token is an upper case form
- Token is in our trigger dictionary
- Token has a numeric character
- Token is an entity
- Token is after "_"
- Token is a stemmed word
- Token frequency

Table 3. Token features for "IL-4 gene regulation involves NFAT1 and NFAT2"

	Token Features						
Token	Upper case	Dictionary	Numeric Character	Entity	After "_"	Stemmed word	Frequency
IL-4 gene	0.25	0	1	1	1	0	1
regulation	0	1	0	0	0	0	1
involves	0	1	0	0	0	0	1
NFAT1	1	0	1	1	0	0	1
NFAT2	1	0	1	1	0	0	1

Table 4 gives the example of candidate in the parser output. In Table 4, the token features are the same features described in Table 2. Column "Bigram dependency" is the sum of the bigrams of candidate token and all its dependency tokens in the parse tree normalized by the total token size. Column "Bigram and trigram of words" is the sum of the bigram and trigram of tokens in the input data normalized by the total token size.

Table 4. Candidate in parser output feature for "IL-4 gene regulation involves NFAT1 and NFAT2"

	Candidate in parser output			
Token	Token features		Bigram dependency	Bigram and trigram of words
	Dictionary	Frequency		
regulation	1	1	0.356	0.524
involves	1	1	0.271	0.445

Dependency chains are built up to the depth of three starting from the token word to be classified. The study uses a set of features for finding the shortest path and

builds a dependency chain considering the three tokens with the shortest path from our candidate token. The sentences are parsed using a dependency parser (Stanford full parser) to identify argument-predicate roles. Words in the sentence and the relationships between these words form the dependency parse tree of the sentence. We use typed-dependency representation to represent our dependency as a simple tuple. For example:

reln(gov, dep)

where *reln* is the dependency relation, *gov* is the governor word and the *dep* is the dependent word. Table 5 shows the features for our shortest path. Table 6 shows the shortest path features for "IL-4 gene regulation involves NFAT1 and NFAT2".

Table 5. Features for the shortest path between candidate trigger and its dependency tokens

- Vertex walk
- Edge walk
- N-gram of dependencies (n= 2, 3)
- N-gram of words (base form + POS) representing governor-dependent relationship (n= 1, 2, 3)

Table 6. Shortest path features for "IL-4 gene regulation involves NFAT1 and NFAT2"

	Shortest path features			
Token	Vertex walk	Edge walk	N-gram of dependency	N-gram of words
regulation	4	3	0.782	0.510
involves	2	1	0.453	0.431

In Table 6, the vertex walk is the sequence of nodes from the candidate node and their substructures. From Figure 2, the vertex walk from the candidate node "*regulation"* has a score of four (three "*theme*" and one "*cause*" respectively). Similarly, the edge walk of the shortest path features accounts for the "*theme*" and "*cause*" pair attributed to each candidate mode. As shown in Figure 2, the candidate node "*regulation*" three pairs attributed to it, and the candidate node "*involves*" has a single pair attributed to it. The N-gram of dependency and N-gram of words are the sums of bigram and trigram of the candidate token and its dependency words in the dependency tree and the tokens in the input sentence normalized by the token size in the background data.

4.4 Edge Detection

The aim of our edge detection approach is to extract event arguments from the semantic graph. Like our trigger detection, our edge detection is based on SVM multiclass classification (SVMlight multi class). Any potential edge in the graph is from an event node to another event node (event nesting) or from an event node to a named entity node. Each edge is classified as a specific argument role such as theme, cause or negative denoting the absence of an edge between the two nodes in the given direction. The role "theme" indicates the target of the event and can be an entity or an event in the case of nested events. The role "cause" refers to the agent of an event. In the case where we predict no theme and cause, we present the role as "negative". It is worth to note that in our edge detection methods, outgoing edges correspond to event arguments. An edge may represent multiple event arguments. Hence, we predict all

edges independently and therefore none also affected by the classification of other edges.

Since we use the Stanford full parser for parsing our input sentences, our feature sets are able to make an extensive use of syntactic dependencies. Our basic idea is to generate features of potential event and event arguments, where event arguments (edges) are the shortest undirected path of syntactic dependencies. Features for candidate tokens of our event edge detector are shown below:

Table 7. Features for candidates of edge detection

Type	Features
Terminal node	Token features listed in Table 2 of terminal node
Three words around the edge pair	N-gram (n=1, 2, 3, 4) of words
Shortest path	Shortest path features listed in Table 5 between terminal nodes Shortest path features listed in Table 5 between the argument trigger and the closest entity

N-grams are defined in the direction of the potential event argument edge. Trigrams are built for each token and its two flanking dependencies, as well as for each dependency and its two flanking tokens. To take into account the varying directions of the dependencies, each pair of consecutive tokens of dependencies forms an additional bigram defining their governor dependent relationship.

Individual component features are defined for each token and edge in a path based on their attributes, which are also combined with the token or edge position at either the interior or the end of the path. We then form edge attributes and combine it

with their direction relative to the path, where our edge attributes constitute features of our terminal tokens.

To create semantic features, we consider attributes of our terminal event or named entity node for all potential argument edges. We create features by considering two key properties. First, the feature is the type of node (event or named entity). Second, if the events or entities have the same head tokens; that is, if there is a self-loop; this implies that we explicitly define the nested arguments as a feature.

Frequency features are also included in the feature set by considering the number of event and named entity nodes per argument in a given sentence. The study considers the presence of the shortest path as a binary feature for a given path.

4.5 Semantic Processing

The graph obtained from the trigger and edge detection methods may contain unwanted nodes that are having event nodes with improper combination of arguments or with event nodes with no arguments. We use the pruning technique to remove those with no arguments. If there exist some event nodes with the same class and the same trigger, they are transformed to the representation by a single node. Events of certain type must have exactly one theme argument and additionally have one cause argument. For these classes, we use a heuristic rule, generating a new event for theme-cause combination. Figure 4 shows one example as below.

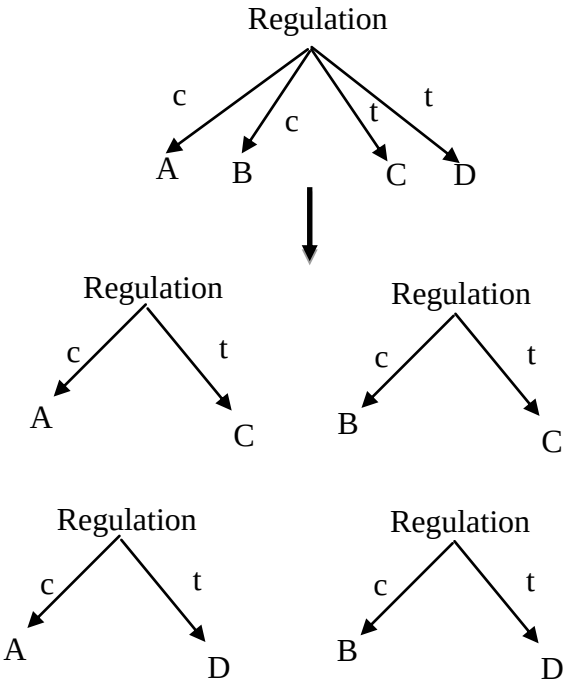

Figure 4. An example of node duplication

In Figure 4, c stands for “cause” and t stands for “theme.”

4.6 Experimental Results and Discussion

4.6.1 Evaluation Metrics

We use the BioNLP-13 shared task’s test evaluation service to test the results of our system. This evaluation service gives recall, precision and F-score of the input regulation network after comparison with the gold standard network for the test data, which ought to be in .a2 file format or SIF file format. In addition, the evaluation service also gives the total number of predictions, matches and deletions.

For evaluation results, recall is the number of correctly aligned slots divided by the number of return slots. Precision is the number of correctly aligned slots divided by

the number of all returned aligned slots. F-score considers both precision and recall rates. The formulas are represented in Equations (2), (3) and (4).

$$\text{Recall (R)} = TP/(TP + FN) \quad \textbf{(2)}$$

$$\text{Precision (P)} = TP/(TP + FP) \quad \textbf{(3)}$$

$$\text{F-score} = 2PR/(P + R) \quad \textbf{(4)}$$

where *TP* is the number of true positives, *FN* is the number of false negatives, and *FP* is the number of false positives.

In the BioNLP-13, the reference is composed of a set of tags representing ground truth. Each tag consists of one or more slots, depending on the tag. A hypothesis is produced by the system comprising another set of tags, each of which also consists of one or more slots. For the test corpus, the scores of participating teams are ranked according to the slot error rate (*SER*) of their results. The formula is as follows.

$$SER = (\textit{total number of slot errors}) / (\textit{total number of slots in the reference}) \quad \textbf{(5)}$$

where the *total number of slot error* is the account of the number of slots in the hypothesis that are align to slots in reference and considered as incorrect. In other word, it is the number of slots in the reference that do not align to any slot in the hypothesis (Makhoul *et al*., 1999).

4.6.2 Result of the Event Extraction Using Graph-Based Feature Sets

We submitted our results in the .a2 file format for evaluation. Our results consist of two parts. First, we check the results of our two components of our system independently. Second, we check the results of the extraction systems of the two components combined.

The experimental results from the first major components are shown in Tables 8. In the table, column “Substitutions” (incorrect slots) is the number of slots in our system that are aligned to slots in the reference and are scored as incorrect. Column “Deletions” (missing slots or false rejections) is the number of slots in the reference that are not aligned to any slot in our system. Column “Insertions” (spurious slots or false acceptances) represents the number of slots in our system that are not aligned to any slot in the reference. Column “Recall” is the number of correctly aligned slots divided by the number of slots. Column “Precision” is the number of correctly aligned slots divided by the number of all returned aligned slots. Column “F-score” considers both precision rate and recall rate. Column “SER” is the score computed with Equation (5).

Table 8. Result from event extraction using graph-based feature sets

Substitutions	Deletions	Insertions	Recall	Precision	F-score	SER
8	62	9	0.2045	0.5143	0.2927	0.8977

5 Semantic Annotation Based on Natural Language Processing

The Result of the Event Extraction Using Graph-Based Feature Sets represented in Section 4 shows nearly about 90 percent SER score. In an attempt to improve this result to get lower SER score, it is observe that the method of "the Event Extraction Using Graph-Based Feature Sets" uses less semantic features and more syntactic features. Hence, the study considers using the method of "Semantic Annotation Based on Natural Language Processing."

Biological sentences are often complex and hard to extract information or knowledge embedded deeply within the sentences without proper inference. In the Semantic Annotation Based on Natural Language Processing approach, three layers of event extraction workflows are used: syntactic analysis, semantic analysis, and inference. The syntactic analysis focuses on extracting the structure in the sentence using the dependency tree from our sentence parsing. The training data is used for semantic analysis and inference.

5.1 Parsing and Preprocessing

Biological events in written literatures are often in a much more complex sentence structures. Extracting events and relations from the complex structures is a difficult task of requiring proper inference. To deal with this complex issue, Stanford dependency parser is used to derive the dependency structure of terms in the input sentence. For the purpose of semantic analysis, semantic tags are assigned to proper nouns of our subjects in a sentence (possible arguments) or to an object. Finally, to

access semantic meanings in a sentence, semantic annotation is used to derive semantic roles of the subject and the object.

5.2 Semantic Annotation

Four different thesauri to assign semantic tags to noun phrases of subjects and objects as described in the work of Huang *et al*. (2006). The thesauri used are MeSH, Gene Ontology (GO), WordNet and a stopword list.[8] We use the following semantic codes shown in Table 9.

Table 9. Semantic codes for annotations

Semantic Code	Description
<ME.D>	Descriptors in MeSH
<ME.C>	Supplementary Concepts in MeSH
<ME.Q>	Qualifiers in MeSH
<SW.A>	Articles in Stopword list
<SW.P>	Preposition in Stopword list
<SW.C>	Conjunction in Stopword list
<SW.N>	Noun in Stopword list
<WN.A>	Adjectives in WordNet
<WN.AD>	Adverb in WordNet
<WN.N>	Noun in WordNet
<WN.V>	Verb in WordNet
<GO.FUN>	Molecular Function in GO
<GO.PRO>	Biological Process in GO
<GO.COM>	Cellular Component in GO

[8] http://jmlr.org/papers/volume5/lewis04a/a11-smart-stop-list/english.stop

The semantic annotation process gives the following result for a particular given sentence in the test data, as shown in Table 10.

Table 10. An example of semantic annotation

Input sentence (from PMID-10075739-S9): We show that GerE binds to two sites that span the -35 region of the cotD promoter.
Results of semantic annotation: <SW.N>We</SW.N> <ME.D>show</ME.D> <SW.P>that</SW.P> <GO.COM>GerE</GO.COM> <ME.D>binds</ME.D> <SW.P>to</SW.P> <ME.D>sites</ME.D> <SW.P>that</SW.P> <WN.N>two</WN.N> <WN.N>sites</WN.N> <ME.D>span</ME.D> <ME.D>region</ME.D> <SW.P>of</SW.P> <SW.P>the</SW.P> <GO.COM>CotD</GO.COM> <ME.D>promoter</ME.D>

5.3 Syntactic Analysis

Syntactic analysis on the sentences can help provide the clues for subsequent semantic interpretation through their syntactic structures. Stanford full parser was used to parse sentences and generate their dependency tree structures. Referring to the study of Huang *et al.* (2006), several rules were applied to extract noun phrases, the real verb of

a sentence, the subject of the sentence, and the object of the sentence from the dependency tree.

For simple cases, we can easily extract the real verb of a sentence containing a single real verb with one or more auxiliaries from the root of a dependency tree. However, the root verb of a sentence is not always the real verb of that particular sentence. In much more complex form, we consider a case where an auxiliary verb connects two verbs in a sentence. To handle such a situation, we traverse the dependency tree to find any node having a verb, which has among its children an auxiliary verb, and such a verb node is selected as our real verb. In other instances, a root verb might be an intransitive verb and may have a clause of a sentence following it. The clause is traversed and searched for other verbs that are present.

To further check if the real verb is a possible event trigger, we identify the subject and the object in the sentence. If a subject and an object relate to the same verb and if the subject or the object of the verb is a named entity, we directly classify our verb in the ontology.

In Figure 5, we get the root verb "*Appears*" first. When traversing the parse tree, the verb "protect" is a child node of "appears" and has an auxiliary verb "to" among its children. Hence, "protect" should be the real action.

In some situations, we have to deal with compound sentences that have the clauses connected by conjunctions, and the relationships between these clauses are determined by the conjunction. We assume that dependent clauses connected by dependent markers such as "after", "although", "as", "because", "if", "since", "until", "when", "whether", and "while" may have causal relationship between them. Clauses joined by the coordinating conjunctions such as "and", "or", "but", "so", and "yet" or joined by independent markers such as "also", "consequently", "however",

“moreover”, and “therefore” are assumed to be independent clauses. These sentences were separated into two clauses and the dependency tree is traversed for extracting the real verb. This real verb presents our identified trigger and we can query our ontology to assign it to a category.

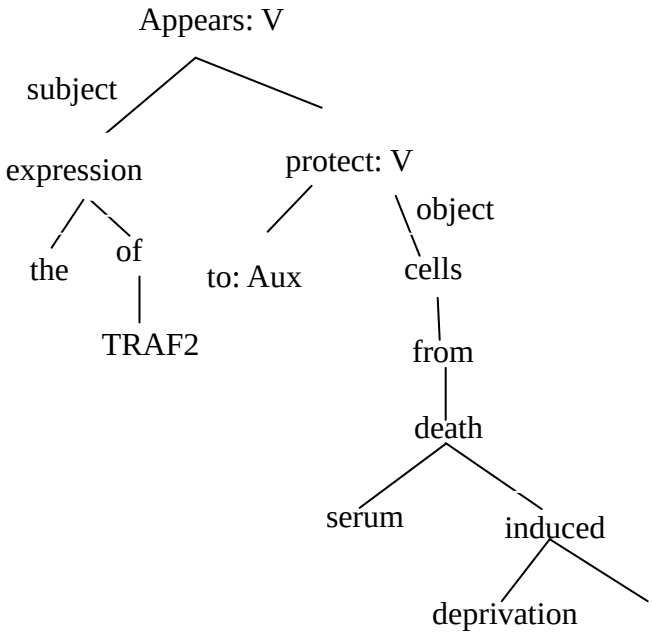

Figure 5. Dependency tree of a sentence “The expression of TRAF2 appears to protect cells from serum deprivation-induced death.”

5.4 Semantic Analysis

In this phase, a dictionary built by grouping annotated triggers and filtering out non-trigger tokens. All the given annotated triggers in the training and development datasets were extracted and categorized based on their event types. For each trigger, the confidence score was calculated by counting the frequency it has being annotated as a trigger and its frequency being found in the training dataset. To decide from our trigger candidates, the confidence score of our SVM classifier was calculated using the distance from the hyperline. We use *n* classes versus all other class approach by having

n classes as a positive class and the rest as a negative class, where *n* ranges from 1 to the total number of event types. We calculate the average distance from the hyperline for each classification to represent our SVM confidence for classification.

A list of prepositions and adjectives derived from the training and development data via dependency parsing forms our list of non-trigger words from the training dataset. Triggers that belong to the non-trigger list as well as those that consist of more than two words are filtered out. To extract the biological events from the parse tree after obtaining a list of candidate triggers, we adapt two syntactic patterns based on similar concepts as found in extraction of PPIs (Protein-Protein Interactions). The study use the pattern for triggers in noun and verb forms.

5.4.1 Triggers in a Noun Form

In extracting events from a *noun phrase* (*NP*), we start with checking if the candidate trigger is a noun. Then we find an *NP*, which is a joined node of this trigger and at least one named entity from the parse tree. The study focuses on two cases: For the first case, we consider a case where the *NP* is follow by a *prepositional phrase* (*PP*) tag. The second case is considered when the trigger is the head of this *NP*. Depending on the trigger type (simple, binding or regulatory event), we extract candidate events using the following two basic rules, called Form 1 and Form 2.

Form 1:

$NP \rightarrow NN\ NN$

Form 2:

$NP \rightarrow NP\ PP$

$NP \rightarrow DT\ NN$

$NP \rightarrow NN\ CC\ NN$

$PP \rightarrow IN\ NP$

where NP means a noun phrase, NN means a noun, PP stands for a prepositional phrase, DT is a determiner, CC means a conjunction, and IN is a preposition. In Form 1 rule, a noun phrase is composed of two nouns. Form 2 rule says that a noun phrase is composed of a noun phrase and a prepositional phrase. The illustration examples of the basic rules are pictured in Figures 6 and 7.

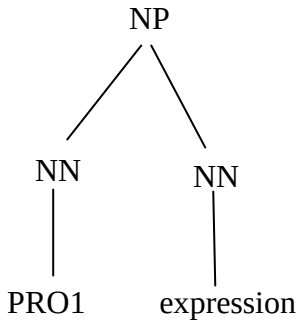

Figure 6. NP pattern example: Form 1

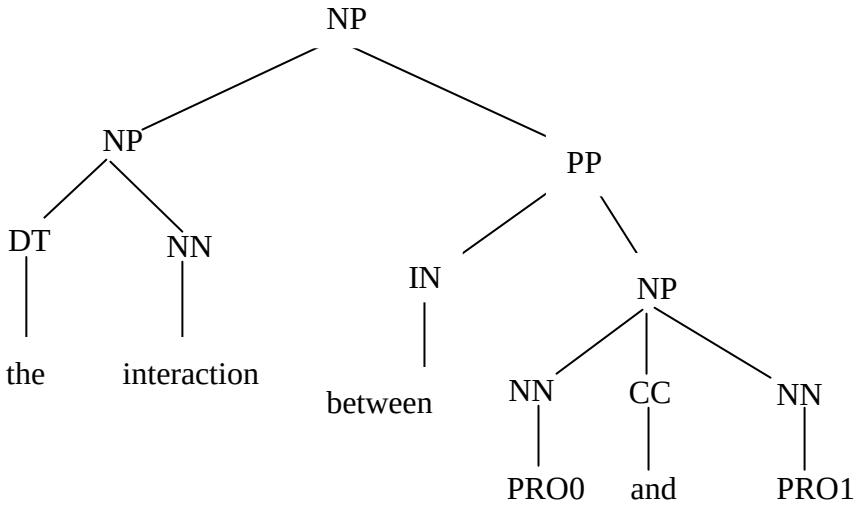

Figure 7. NP pattern example: Form 2

In Figures 6 and 7, PRO0 and PRO1 stand for some named entities.

For the simple or regulatory event type, if the *NP* does not have a *PP* tag, we extract all proteins on the left of the trigger from the *NP* and form event pairs <*trigger*, *argument*>. Otherwise, we extract all proteins on the right of the trigger from the NP and form event pairs <*trigger*, *argument*>. For example, <expression, PRO1> is an extraction from Form 1 in Figure 6. For the binding event type, if the *NP* does not have a *PP* tag, and if proteins are in a compound form such as *PRO0/PRO1*, *PRO0-PRO1*, where PRO0 and PRO1 are entities, then form an event triple <*trigger*, *argument1*, *argument2*>. Otherwise, form events pairs <*trigger*, *argument*>. If an *NP* has a *PP* tag, which is one of the following preposition pairs, *between/and*, *of/with*, *of/to*, then split the proteins from *NP* into two lists based on the *PP* or *Coordinating Conjunctions* (*CC*). Form triples <*trigger*, *argument1*, *argument2*> where *argument1* and *argument2* are from different lists. For example, <*interaction, PRO0, PRO1*>, as shown in From 2 of Figure 7.

5.4.2 Triggers in a Verb Form

To extract events from a *verb phrase* (*VP*), the trigger is checked if it is a verb. If it is, then the next step is to find a *VP*, which is a direct parent of this trigger from the parse tree, and to find a sister *NP* immediately preceding this *VP*. In the following, we extract the candidate events as in the *NP* case mentioned above.

5.5 Experimental Results and Discussion

5.5.1 Result of semantic annotation based on natural language processing

As in our Event Extraction Using Graph-Based Feature Sets approach, the submitted format is in the .a2 file format for evaluation. The experimental result is shown in Table 11. From Tables 8 and 11, it is observable that the score of Event Extraction Using Graph-Based Feature Sets approach is slightly different from the score of Semantic Annotation Based on Natural Language Processing. It is worth to notice that, the Event Extraction Using Graph-Based Feature Sets approach has better SER score because its total slot error is lower. Here, the slot error total is the sum of substitution, deletion and insertions scores.

Table 11. Result of semantic annotation based on NLP

Substitution	Deletions	Insertions	Recall	Precisions	F-score	SER
9	61	10	0.2045	0.4865	0.288	0.9091

In our stacking and filtering component of our system, we concentrate on the important event extraction in the GRN-shared task. Event extraction is a primary objective in the formation of a regulation network. Every biological molecule or genetic entity is associated to a molecular or a genic event. Hence, it is essential to be able to extract biological events and their arguments from any given input sentence.

In our attempt to improve the precision rate and lower the slot error of the system, we combine the output results of event extraction from the two major components of our system. For a given sentence, we check into the output result on the extracted events and their arguments for similarities and differences. For an event with similar

arguments that appear from the two components, the approach retains a single case. In the case where the similar event extracted from the two components but which has different arguments, we keep both cases of the events. The stacking and filtering component functions to combine the two results of extracted events with their respective arguments and removing redundancies. The evaluation score of this result using the shared task's test evaluation service is given in Table 12 below.

Table 12. System score from the shared task evaluation service with the combined strategy

Substitution	Deletion	Insertions	Recall	Precision	F-score	SER
9	59	10	0.2273	0.5128	0.3150	0.8864

From Table 12, it can be understood that our combined result is dominate by the approach of event extraction using the graph-based feature sets. However, the result of individual components of our system is very much similar. Our semantic annotation approach has a poor score in the number of deletions causing a higher slot error rate.

6 Statistical Approach for Biological Event Extraction Using Markov's Method

From the above three results: result of Event Extraction Using Graph-Based Feature Sets approach, result of Semantic Annotation Based on Natural Language Processing approach, and the combined result, it is worth to notice the slight differences among the three results. The best score is obtained from the combined result of the two approaches. However, the difference in SER score is not significant enough. In an attempt to a get better result, it is observable the above two approaches in Section 4 and Section 5 do not focus on statistical analysis in combinational linguistics. In an attempt to get a better result, the study uses the statistical analysis using Markov's Logic.

Markov Logic (Richardson and Domingos, 2006) is a statistical relational learning language based on First Order Logic and Markov Networks. A formalism that extends First Order Logic to allow formulae that can be violated with some penalty. From an alternative point of view, an expressive template language uses First Order Logic formulae to instantiate Markov Networks for repetitive structures.

We can introduce Markov Logic by considering the event extraction task (as relational structure over token). In Markov Logic, we can model this task by introducing logical predicates such as *event*(*event_token, type*) and *argu*(*role*, *role_filer*). We achieve this by using profile Hidden Markov's Model (profile HMM). Profile HMM has been well adopted for sequence alignment in computational biology. However, instead of aiming to identify highly conserved region/motifs of multiple alignments, our profile analysis aims at extracting patterns of occurrence of events and their arguments in training and development data and uses such information as

background observation. Then we specify a set of weighted First Order formulae that define a distribution over sets of the ground atoms of these predicates (so-called possible worlds). Note that we will refer to word token as "*observed* " because they are already known in advance. In contrast, annotation and annotation types are as "*hidden*" because we also need to infer its ground atoms at test time.

Figure 8 shows the flowchart of the statistical approach for biological event extraction using Markov's method. The details are in the following subsections.

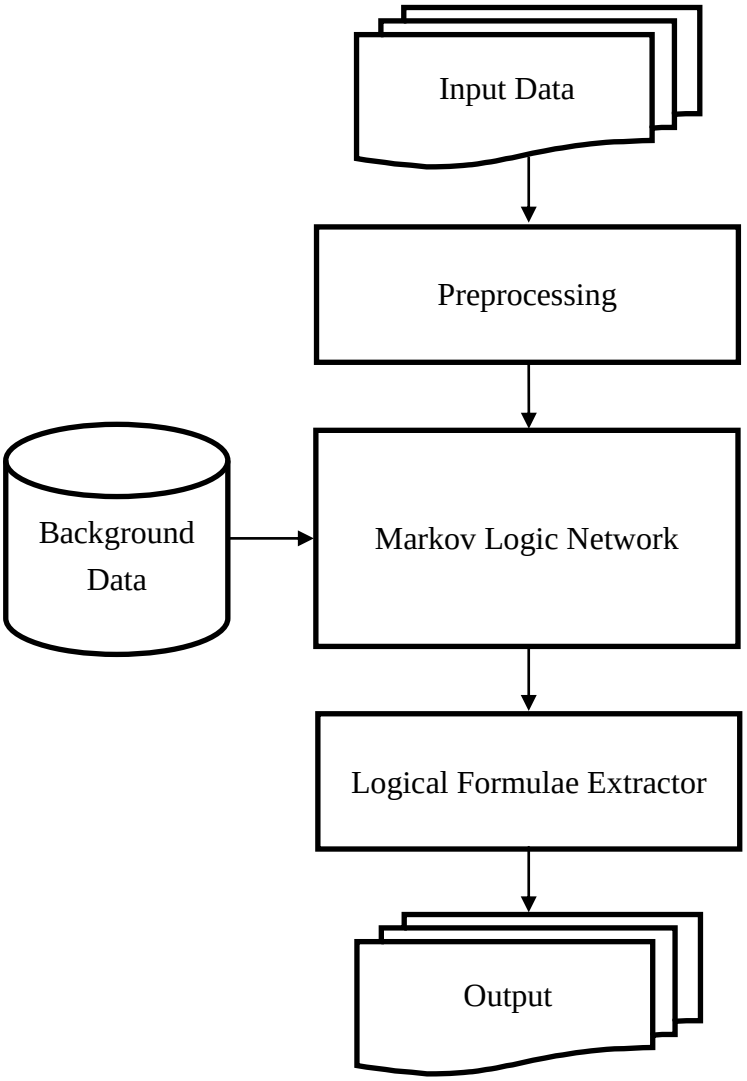

Figure 8. Statistical Approach for Biological Event Extraction Using Markov's Method

6.1 Preprocessing

In the preprocessing level, we take into consideration the given annotation of the training data and the development data. The first approach in the preprocessing is the extraction of the annotations in the training and development data in logical predicate format. The objective of this is to represent event structures as relational structures over tokens. For each input test file, we record the three top text documents in the training and development data that have a close similarity with the input in the test file.

In the next approach of preprocessing, the objective is to obtain any term dependency in the input text data from the shared task's test files. This is achieved by parsing the input sentences of the test data using Stanford parser. This helps directly deal with word tokens instead of character offset or a sentence token and be able to trace dependency relationship between tokens of interest. It further enables to formulate the probabilistic model in terms of consistent word tokens.

6.2 Markov's Logic Network for Event Prediction

For each test file given in the shared task, a link of a network is developed to connect it to all the three nearest similar documents in our background data using cosine similarity. This aims at assigning probabilistic weights to the possible event, event triggers, and event argument in the test file. We use a case prediction of entities found in the test data within the dataset created in the preprocessing level. The proposed approach defines a predicate *entity*(*i*) to represent an entity token *i* in the background data. For each entity in the test data, if it is found in the dataset, it is assigned a value of one (true), otherwise assigned zero (false). For any entity with value one assignment, a query for all its associated annotations such as events, relations and all other

associated information that is a target of the shared task in our background data using predicates is excuted. These annotations are assigned a probabilistic value that describes the probability of its appearance in relation to a particular entity within annotations found in our dataset.

From the created dataset, a set of logical predicates such as *eventType*(*event_token*, *type*), and *fun*(*argu*(), *role*) are defined. These predicates assign the weighted scores depending on the appearance of the arguments within the predicates in the processed test input data. The study converts the task into the link prediction over a sequence of predicates describing tokens.

For simplicity, we introduce a formal representation of the events in a sentence and identify the token position in the sentence. We can describe an event *e* as a tuple (*i*, *t*, *fun*()). Here *i* is the token position of the trigger word, *t* is the event type of *e* and *fun*() is a set of label arguments (*a*, *r*) where each *a* is either an entity, or event, and *r* is role *a* play in with respect to *e*. For the shared task, five primary annotations are given: *text annotation, event annotation, event modification annotation, relation annotation* and *normalization annotation*. The *text annotation* defines an entity or event trigger in a specific span of texts and assigns it a type. *Event annotation* gives annotations of events and relations. Event annotations are a primary extraction target in the main tasks of BioNLP-2013 shared task. Gold standard event annotations are given in both training and development data. Participants are required to create both the event trigger and event annotation for the test data. *Event modification annotations* contain additional information about other annotations, such as events that are stated speculatively or in a negative context. *Relation annotations* are similar to *event annotations* but defined by arguments and types. The *normalization annotation* gives an external reference to other annotations. In the test data, only the entity and the text

files containing the abstracts are given. For each sentence, we also introduce a state tuple, that represents the sentence as an event structure for a particular sentence represented using a predicate. An example is shown as below:

start (*event*, *eventType*, *eventTrigger, eventArgument)*

For our test data, we make efforts to effectively use both the given text files and the annotation files. Entities in the annotations are assigned binary values during the preprocessing. For each entity in the test file, if and only if the entity is in the background, then a relational link is developed between the two corresponding files. We then create a vector to contain any event or event trigger functionally related to the entity under consideration.

We assign to each event or event trigger in the background vector, *v*, a weight, *w*, as follow:

$$w = f_e / |x| \quad \textbf{(6)}$$

where f_e is the frequency of event *e* in test file and $|x|$ is the token count of test file *x*. Supposed an entity *e* has all background vector elements having $w \geq 0$, that event state is considered an event case for the corresponding input test file. Furthermore, we use a dependency parser to parse each input test sentence and find the dependency relationship between tokens. We use a predicate *dep*(*i, j, d*), where *i* is the head of token *j* and *d* is the dependency type. Next, we develop Markov's Logic Network and develop logical formulae to extract *events, event triggers, event arguments* and other annotations.

6.3 Extraction of Events Using Logical Formulae

Our logical predicates can be classified into "*hidden"* or "*observed"* predicates. With our "*hidden"* and "*observed"* predicates, we predict the events and other annotations using logical formulae considering information both in our background data and in test data. Our logical formula includes the following:

Table 13. Logical formulae for event extraction

Logical Expression	**Description**
$\exists i_t$ (*background_data.entity*(i_t) $\rightarrow \exists r$ *background_data.role*(i_t, r))	If there is any test entity i_t that occurred in the background data, then there is a role r taken by i_t .
$\exists r$ *role*(r) $\rightarrow \exists$ i (*event*(i) $\vee$ *eventTrigger*(i))	If there exists a role, then there must exist an event or event trigger filing the role.
$\exists i$ (*event*(i) $\rightarrow \exists t$ *eventType(i, t)*)	If there exists event i, then there is an event type t describing event i.
$\exists i_t$ *entity*(i_t) $\wedge \exists e_t$ $\exists t$ (*event*(e_t) $\in$ *background_data.event*() $\wedge$ *eventType*(t)) $\rightarrow$ (*state* $\leftarrow$ *background.annotation*)	If an entity i_t with weight $w \geq 0$ has event e_t found in the background data, add the background annotation to the test annotation.
$\forall e$ $\exists t$ $\forall x$ (*eventType*(e, t) $\wedge$ ($t \neq x$) $\rightarrow$ $\neg$ *eventType*(e, x))	Every event e must have only one event type.
$\forall i$ $\forall j$ $\forall k$ $\forall r$ ((*index*(i) < *index*(j)) $\wedge$ (*index*(j) < *index*(k)) $\wedge$ *role*(i , j, r)) $\rightarrow$ $\neg$ *role*(i, k, r)	We cannot have roles within roles.
$\forall i$ $\forall j$ $\forall r$ $\forall s$ (*role*(i, j, r) $\wedge$ ($r \neq s$) $\rightarrow$ $\neg$ *role*(i, j, s))	There can only be a single role assignment.

6.4 Experimental Results and Discussion

Our result for the statistical approach is much better than the first two approaches mentioned above. The evaluation result shown as in Table 14. Comparing to the approaches in Sections 4 and 5, the statistical approach for biological event extraction using Markov's method approach has a lower slot error with better precision and recall.

Table 14. System score from the shared task evaluation service with the statistical approach

Substitutions	Deletion	Insertion	Recall	Precision	F-Score	SER
0	56	6	0.3086	0.8065	0.4464	0.7654

As shown in the above table, our statistical approach has the best performance among our three approaches to event extraction of biological events.

7 Conclusion

In this thesis work, we have implemented the combined semantic and syntactic approach, and a statistical approach based on Markov's logic to extract biological events from literatures. From this result, we form a gene regulation network for the GRN shared task. The study uses the test evaluation system provided by the BioNLP-2013 shared task community to test our result for the gene regulation network.

Statistical methods in natural language processing have been extensively used to solve quite a lot of problems. Several statistical approaches have proven to achieve good results. Markov's logic has been used in the speech recognition problem and has achieved a great success in this area. In our approach, we try to show that such a statistical approach as Markov's Logic can also be extended to solve for biological event extraction. We show that, with our different results, Markov's logic approach has a very good SER score and about 80% precision in forming our gene regulation network.

Comparing our results to results of participants in the BioNLP shared task 2013 as shown in Table 15, our systems achieves better SER score than certain participants.

Table 15. Results of participants in BioNLP shared task 2013

Participant	Rank	SER	Recall	Precision	F1
University of Ljubljana	1	0.73	0.34	0.68	0.45
K.U.Leuven	2	0.83	0.23	0.50	0.31
TEES-2.1	3	0.86	0.23	0.54	0.32
IRISA-TexMex	4	0.91	0.41	0.40	0.40
EVEX	5	0.92	0.13	0.44	0.19

Our evaluation result could be ranked 0.76 and 0.88 for our statistical approach and the combined graph-based and semantic approach respectively among the ranking of participating teams that took part in BioNLP-2013 shared task competition.

Our future work will further focus on more into the application of Markov's methods and other statistical approaches in computational linguistics. Mathematical modeling has played a significant role in many research areas. We believe it can be applied to extraction of biological events from any given biological structure. Statistical Natural Language Processing includes several quantative approaches such as probabilistic modeling, information theory, and linear algebra. We believe a careful study of this approaches can be of great interest in the area of event extraction.

References

BioNLP Shared Task. Available from http://2013.bionlp-st.org/

Björne, Jari, Heimonen, Juho, Ginter, Filip, Airola, Antti, Pahikkala, Tapio and Salakoski, Tapio (2011). Extracting Contextualized Complex Biological Events with Rich Graph-based Feature Sets, Computational Intelligence, 27 (4), pages 541–557, 2011.

Bui, Quoc-Chinh and Sloot, M.A. Peter (2011). Extracting Biological Events from Text Using Simple Syntactic Patterns, Proceedings of BioNLP Shared Task 2011 Workshop, pages 143–146, 2011.

Gene Ontology. Available from http://www.geneontology.org/

Huang, Yu-Ting, Yeh, Hsiang-Yuan, Cheng, Shih-Wu, Tu, Chien-Chih, Kuo, Chi-Li and Soo, Von-Wun (2006). Automatic Extraction of Information about the Molecular Interactions in Biological Pathways from Texts Based on Ontology and Semantic Processing, IEEE International Conference on System, Man, and Cybernetics, pages 3679–3684, 2006.

Makhoul, John, Kubala, Francis, Schwartz, Richard and Weischedel, Ralph (1999). Performance Measures for Information Extraction, Proceedings of DARPA Broadcast News Workshop, pages 249–252, 1999.

Martinez, David and Baldwin, Timothy (2011). Word Sense Disambiguation for Event Trigger Word Detection in Biomedicine, Proceedings of the Fourth International Workshop on Data and Text Mining in Biomedical Informatics (DTMBio 2010), BMC Bioinformatics 2011, 12 (Suppl 2): S4, 2011.

McClosky, David, Riedel, Sebastian, Surdeanu, Mihai, McCallum, Andrew and Manning, D. Christopher (2012). Combining Joint Models for Biomedical Event Extraction, BMC Bioinformatics 2012, 13(Suppl 11): S9, 2012.

McClosky, David, Surdeanu, Mihai and Manning, D. Christopher (2011). Event Extraction as Dependency Parsing, Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 1626–1635, 2011.

MeSH. Available from http://www.ncbi.nlm.nih.gov/mesh

PubMed. Available from http://www.ncbi.nlm.nih.gov/pubmed

Richardson, Matthew and Domingos, Pedro (2006). Markov Logic Networks, Machine Learning, Volume 62, pages 107–136, 2006.

Riedel, Sebastian, Sætre, Rune, Chun, Hong-Woo, Takagi, Toshihisa and Tsujii, Jun'ichi (2011). Bio-Molecular Event Extraction with Markov Logic, Computational Intelligence, 27 (4), pages 558–582, 2011.

Stanford full parser. Available from http://nlp.stanford.edu/software/lex-parser.shtml

Stopword List. Available from http://jmlr.org/papers/volume5/lewis04a/aa11-smart-stop-list/english.stop

SVMLight Multi-class. Available from http://www.cs.cornell.edu/people/tj/svm_light/svm_multiclass.html

Tsochantaridis, Ioannis, Hofmann, Thomas, Joachims, Thorsten and Altun, Yasemin (2004). Support Vector Machine Learning for Interdependent and Structured

Output Spaces, Proceedings of the Twenty-first International Conference on Machine Learning (ICML'04), ACM, Banff, Canada, pages 104–111, 2004.

WordNet. Available from http://wordnet.princeton.edu/