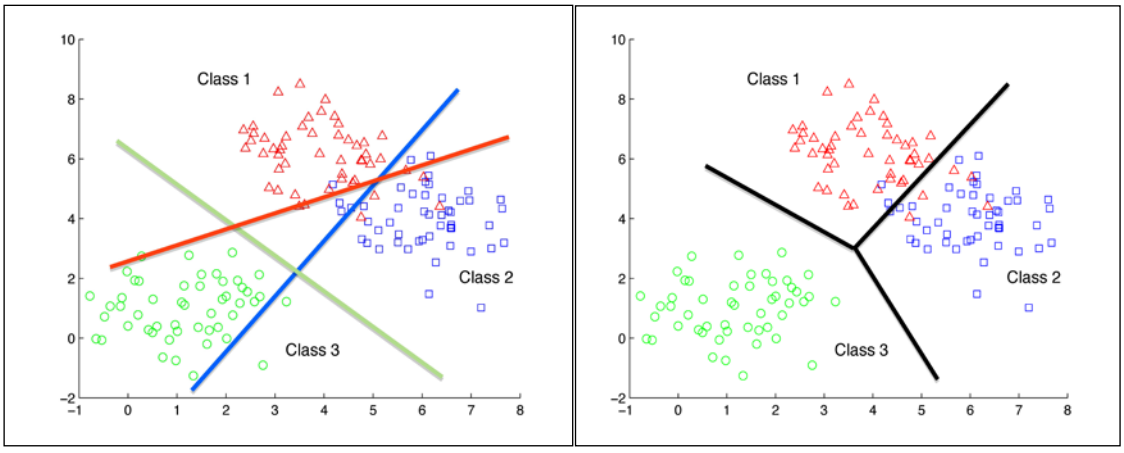
Fast recursive biomedical event extraction

Xiao Liu

Par **Xiao LIU**

Fast recursive biomedical event extraction

Thèse présentée
pour l'obtention du grade
de Docteur de l'UTC



Soutenue le 25 septembre 2014
Spécialité : Technologies de l'Information et des Systèmes :
Unité de recherche Heudyasic (UMR-7253)

D1963

Université de Technologie de Compiègne

Doctoral Thesis

Fast Recursive Biomedical Event Extraction

Author:
Xiao Liu

Supervisors:
Antoine Bordes
Yves Grandvalet

Spécialité : Technologies de l'Information et des Systèmes

Unité de recherche Heudyasic (UMR-7253)

Thèse présentée pour l'obtention du grade de docteur de l'UTC

25/09/2014

"The question of whether Machines Can Think... is about as relevant as the question of whether Submarines Can Swim."

Edsger W. Dijkstra

Résumé

L'internet et les nouvelles formes de média de communication, d'information, et de divertissement ont entraîné une croissance massive de la quantité des données numérique. Le traitement et l'interprétation automatique de ces données permettent de créer des bases de connaissances, de rendre les recherches plus efficaces et d'effectuer des recherches sur les médias sociaux. Les travaux de recherche sur le traitement automatique du langage naturel concernent la conception et le développement d'algorithmes, qui permettent aux ordinateurs de traiter automatiquement le langage naturel dans les textes, les contenus audio, les images ou les vidéos, pour des tâches spécifiques. De par la complexité du langage humain, le traitement du langage naturel sous forme textuelle peut être divisé en 4 niveaux : la morphologie, la syntaxe, la sémantique et la pragmatique. Les technologies actuelles du traitement du langage naturel ont eu de grands succès sur les tâches liées aux deux premiers niveaux, ce qui a permis la commercialisation de beaucoup d'applications comme les moteurs de recherche. Cependant, les moteurs de recherches avancés (structurels) nécessitent une interprétation du langage plus avancée. L'extraction d'information consiste à extraire des informations structurelles à partir des ressources non annotées ou semi-annotées, afin de permettre des recherches avancées et la création automatique des bases de connaissances.

Cette thèse étudie le problème d'extraction d'information dans le domaine spécifique de l'extraction des événements biomédicaux. Nous proposons une solution efficace, qui fait un compromis entre deux types principaux de méthodes proposées dans la littérature. Cette solution arrive à un bon équilibre entre la performance et la rapidité, ce qui la rend utilisable pour traiter des données à grande échelle. Elle a des performances compétitives face aux meilleurs modèles existant avec une complexité en temps de calcul beaucoup plus faible. Lors la conception de ce modèle, nous étudions également les effets des différents classifieurs qui sont souvent proposés pour la résolution des problèmes de classification multi-classe. Nous testons également deux méthodes permettant d'intégrer des représentations vectorielles des mots appris par apprentissage profond (*deep learning*). Même si les classifieurs différents et l'intégration des vecteurs de mots n'améliorent pas grandement la performance, nous pensons que ces directions de recherche ont du potentiel et sont prometteuses pour améliorer l'extraction d'information.

Abstract

Internet as well as all the modern media of communication, information and entertainment entail a massive increase of digital data quantities. Automatically processing and understanding these massive data enables creating large knowledge bases, more efficient search, social medial research, *etc.* Natural language processing research concerns the design and development of algorithms that allow computers to process natural language in texts, audios, images or videos automatically for specific tasks. Due to the complexity of human language, natural language processing of text can be divided into four levels: morphology, syntax, semantics and pragmatics. Current natural language processing technologies have achieved great successes in the tasks of the first two levels, leading to successes in many commercial applications such as search. However, advanced structured search engine would require computers to understand language deeper than at the morphology and syntactic levels. Information extraction is designed to extract meaningful structural information from unannotated or semi-annotated resources to enable advanced search and automatically create knowledge bases for further use.

This thesis studies the problem of information extraction in the specific domain of biomedical event extraction. We propose an efficient solution, which is a trade-off between the two main trends of methods proposed in previous work. This solution reaches a good balance point between performance and speed, which is suitable to process large-scale data. It achieves competitive performance to the best models with a much lower computational complexity. While designing this model, we also studied the effects of different classifiers that are usually proposed to solve the multi-class classification problem. We also tested two simple methods to integrate word vector representations learned by deep learning method into our model. Even if different classifiers and the integration of word vectors do not greatly improve the performance, we believe that these research directions carry some promising potential for improving information extraction.

Contents

List of Figures

List of Tables

Abbreviations

NLP	**N**ature **L**anguage **P**rocessing
NER	**N**amed **E**ntity **R**ecognition
IE	**I**nformation **E**xtraction
POS	**P**art **O**f **S**peech
PTB	**P**enn **T**ree**B**ank
BOW	**B**ag **O**f **W**ord
SVM	**S**upport **V**ector **M**achine
LR	**L**ogistic **R**egresion
MNLogit	**M**ulti-**N**omial **L**ogistic regression
SVT	**S**ingle e**V**en**T**
BIND	**BIND**ing event
REG	**REG**ulation event
Non-REG	**Non-REG**ulation event

Symbols

Task Definitions

TRIGGER	character chain acting as an anchor of event
THEME	main participant of event, formulated as argument of TRIGGER
THEME_2	optional second participant of event
CAUSE	cause of event

Modelizations

S	Sentence index
$\mathcal{T}_S$	set of candidate entities for TRIGGER in sentence S
$t \in \mathcal{T}_S$	candidate entity for TRIGGER
$\mathcal{A}_S$	set of candidate arguments in sentence S
$a \in \mathcal{A}_S$	candidate argument
$\mathcal{P}_S$	set of candidate (t, a) pairs in sentence S
$p \in \mathcal{P}_S$	candidate (t, a) pair
$\mathcal{E}_S$	set of events in sentence S
$e \in \mathcal{E}_S$	event
$\mathcal{Y}$	set of categories of events

Chapter 1

Introduction

This thesis introduces an efficient way of extracting biomedical information from text, which was applied to the BioNLP Genia task. In this chapter, we illustrate the background of this thesis, the details of a focused task and give a concise summary of our achievements. Section 1.1 proposes a prospect of information extraction and shows the difficulty to work in a specific domain. After that, Section 1.2 summarizes the contributions that we have been developing for the BioNLP Genia task. The final section (Section 1.3) briefly describes the following chapters.

1.1 Biomedical Information Extraction from Text

In this section, we introduce the domain of the task we address. The BioNLP Genia task consists of extracting biomedical events, following predefined formulas, from raw text. Besides the BioNLP Genia task [1–3], there are many other biomedical event extraction tasks [4–13] that defined different events to extract such as the BioNLP gene regulation ontology task [4], the BioNLP cancer genetic task [5], the BioNLP pathway curation task [6], *etc.* All of these tasks intend to extract structured information from raw text, and hence fall into the range of *information extraction* from text. Information extraction from text usually relies on natural language processing (NLP), and is always discussed as a high-level NLP task. Before introducing the biomedical information extraction task, we define NLP in Section 1.1.1 and the general problem of Information Extraction in Section 1.1.2

1.1.1 Natural Language Processing

Natural language processing concerns the interaction between computers and human language. Two main categories of challenges are involved in natural language processing: one is natural language understanding; the other is natural language generation. These two tasks consider opposite directions since natural language understanding aims to transform the input sentence into machine representation whereas natural language generation determines how, for a machine, to express concepts in human language. Natural language understanding is now widely used in search engines, text summarization,

voice control, document categorization, large-scale content analysis, *etc.* From a commercial perspective, natural language understanding is more mature than natural language generation that is usually used as a part of summarization or machine translation applications. For the long-term goal, which targets to generate human-like sentences, current natural language generation systems are far from being effectively used. We introduce below some NLP tasks devoted to natural language understanding.

1.1.1.1 Tasks of NLP

Natural language processing is a big domain that addresses many tasks, where some of them have direct real-world applications, whereas others are sub-tasks that are used to aid in solving the larger tasks. All the tasks have well-defined problem settings, standard metrics to evaluate the solutions and standard corpora on which these solutions can be evaluated.

Real-World Tasks We start by introducing the tasks directly related to real-world applications. Natural language can be carried out in different resources such as texts, voices, images, *etc.* The vast majority of NLP tasks focuses on the processing of texts while others are often grouped into multimedia NLP tasks. Typical multimedia NLP tasks include speech recognition (voice to text), optical character recognition (image to text) and speech generation (text to voice), which are transformations between texts and other media formats. Advanced processing of natural language carried out in the multimedia resources are usually implemented by combining advanced textual processing and transformation between multimedia resources and texts.

Many tasks are proposed for texts like automatic summarization, machine translation, question answering, sentiment analysis, information retrieval, *etc.* General search engines return documents relying on information retrieval technology and snippets of these documents generated by automatic summarization. Question answering aims at building systems that "understand" the questions asked by users in human language and return the result by searching a structured knowledge base. Machine translation aims to translate the text or speech from one language to another. Human translation process basically contains two steps: 1) decoding the meaning from source text; 2) re-encoding the meaning in target language. Current machine translation method works in a similar way, which involves both natural language understanding and natural language generation. Sentiment analysis extracts subject information from document sets and is used to track the public opinion from social media for marketing or political use.

Inter-media Tasks Natural language is a highly abstract representation of meanings and is hard to be processed directly by a computer. Similar to image processing, all the real-world NLP tasks require plenty of refined heuristic representations of original texts to get a good performance. Following the linguistics, which studies the structure and rules of human languages, many sub-tasks were proposed to construct different representations of sentences from the simplest morphology of words to complex specific structures. We group these sub-tasks into three categories: 1) segmentation, 2) tagging, 3) structured extraction.

The segmentation tasks aim to split continuous speech or text into predefined linguistic units. The sub-task for speech recognition task is speech segmentation, which identifies

the boundaries of words, syllables or phonemes in a soundtrack. Other segmentation tasks segment specific units from texts, where sentence breaking identifies the boundaries of sentences in documents, word segmentation breaks the sentences into words, morphological segmentation identifies the individual morphemes in words. Note that morphological segmentation is rarely used in high-level tasks while sentence breaking and word segmentation are almost systematic in all the NLP tasks.

We conclude by tagging tasks that sequentially identify the elements in sentences like part-of-speech tagging, word sense disambiguation and named entity recognition. The part-of-speech tagging task requires determining the part of speech (POS) for each word given a sentence. This task needs to choose the right POS while one word can serve as multiple parts of speech. It is similar to word sense disambiguation, which selects the meaning which makes the most sense in context when words have more than one meaning. In practice, the POS task is much easier than word sense disambiguation. Unlike marking the classes of words, named entity recognition aims to recognize the classes of character chains, which are not necessary a word.

The structured extraction tasks are usually designed to extract specific relations between predefined entities, which can be words or named entities. Coreference resolution, parsing, relationship extraction fall into this category. Coreference resolution are proposed to recognize the words that refer to the same objects. One of these tasks is anaphora resolution, which concerns the match of pronouns to the nouns or names that they refer to. Parsing tasks provide a grammatical analysis represented by a parse tree or graph. Relationship extraction is very similar to parsing except that the specified relations depend more on semantics than on syntactic and their targets are often restricted named entities instead of all the words. Parsing is often used in other structured tasks as an essential feature while coreference resolution and relationship extraction are used to enrich the performance of real-world applications.

The output space in structured extraction is much larger than for other tasks. Besides, except syntactic parsing, the state-of-the-art performance of structured extraction is far below human performance. Despite the complexity brought by structured output, structured extraction tasks are more difficult because these tasks heavily rely on larger contexts and deeper semantic comprehension of sentences. Moreover, structured extraction tasks need the support from many other inter-media tasks, which leads to error cascading problems.

1.1.1.2 Methods of NLP

There are two main streams of methods for NLP: the first are rule-based systems, the second are statistical models. The rule based systems were originally developed by people willing to create machines that can pass the Turing test. Programmers manually created ontologies, which are structured databases that represent the real-world information and can be understood by computers, along with the rulers to generate sentences and translate between two languages. The rule based systems want to simulate the logic of humans, but they are hard to extend since the rules have to be written by experts and strict logics are short to handle the ambiguity of human language and faults. Integrating machine learning methods into rule-based systems enables the system to discover new rules or knowledge by itself given new corpus. But their performances are usually worse than statistical models in practical applications.

In the past three decades, researchers focused on statistical models, which achieved great successes in many commercial applications. Statistical models make soft probabilistic decisions given by real-valued functions, which predict the outcome by weighting the features of the input data. The weights are learned from training corpus via machine learning methods. Such models are more robust when given new examples, especially when inputs contain errors, which is very common in real-world data.

For the tasks introduced in previous section, the simple segmentation tasks of some languages like English are solved by rule-based methods since their variations are regular and easy to treat. But for tagging tasks and structured extraction tasks, statistic models generally work better than rule-based models. Therefore, statistical models are increasingly popular in commercial applications and research.

1.1.2 Information Extraction from Text

Information extraction aims to extract structured information from unstructured or semi-structured resources. This technology permits to automatically construct database that can be used in precise structured information retrieval, knowledge base construction, *etc.* The target resources can be human language text, audio or even video; but most of efforts were made on processing text through natural language processing (NLP). The problem tackled in this thesis falls into the first kind. Information extraction tasks usually require to output graphs or trees, which are very similar to the ones of syntactic parsing tasks. However, syntactic parsing tasks emphasize on syntactic relations whereas information extraction tasks emphasize on semantic relations. Besides, syntactic parsing tasks do not consider the entities, but only the words in sentences. Hence, information extraction tasks are much more difficult than parsing.

With the widespread use of Internet, massive information is provided by users in unorganized texts. This huge amount of data contains vast of valuable information that people would like to retrieve for many usages. For example, current search engines mainly reply on keywords searches, which retrieve documents with respect to the importance of keywords and meta data in web pages. However, the high-quality meta data is not always available and is not detailed enough for some search problems that require refined and deep comprehension of documents. Information extraction allows to create logical form indices from raw text, which are necessary for precise search with logical restrictions. Furthermore, reliable information extraction can automatically generate knowledge base, which is essential for many tasks like expert systems.

Typical sub-tasks of information extraction include: named entity recognition, relationship resolution and coreference resolution. In the information extraction process, named entity recognition is used to find the target entities, relationship resolution is the main part that creates the desired structures, which are often graphs or trees. Coreference resolution is used to reinforce the performance of information extraction since the anaphoric links are very useful to extract potential relations. In these sub-tasks, almost all the other inter-media tasks are involved to provide features in actual approaches.

TABLE 1.1: BioNLP Genia event extraction task definitions. "?" means that this argument is optional, "+" means the this argument can occur more than once.

Event Type	Type of Theme participant, other participants (type)	Additional Arguments
Gene_ expression	Protein	
Transcription	Protein	
Protein_ catabolism	Protein	
Phosphorylation	Protein	Site (Entity)?
Localization	Protein	AtLoc (Entity)?, ToLoc (Entity)?
Binding	Protein+	Site(Entity)+
Regulation	Protein/Event, Cause (Protein/Event)?	Site (Entity)?, CSite (Entity)?
Positive_ regulation	Protein/Event, Cause (Protein/Event)?	Site (Entity)?, CSite (Entity)?
Negative_ regulation	Protein/Event, Cause (Protein/Event)?	Site (Entity)?, CSite (Entity)?

1.1.3 BioNLP Genia Task

BioNLP Genia task is a major task of the BioNLP Shared Task started in 2009. It concerns the recognition of bio-molecular events that appear in biomedical literature. The definition of bio-event is broadly described as a change on the state of a bio-molecular or bio-molecules. In BioNLP Shared Task 2011 and 2013, more tasks were added (see Nédellec et al. [7]) and the definition of the Genia task changed in 2013: new event types have been added into the framework and integrated coreference resolution becomes a part of this task. This task focuses on extraction of bio-events particularly on proteins or genes, where proteins and genes are not distinguished and given. It was divided into three sub-tasks:

1. **Core event extraction** task requires to identify the event triggers, their types and the participants of events.
2. **Event enrichment** task requires to extract additional arguments to enrich the core events, such as the location of events. One has to recognize entities of the additional arguments and detect the relations between these entities and the core events.
3. **Negation and speculation recognition** task requires to find negations and speculations regarding events extracted by task 1.

Table 1.1 lists the event definitions used in BioNLP 2009 and 2011 Genia tasks. To be consistent with the nomenclature of this thesis, we used *participants* instead of *Primary Arguments* in [1, 2] and used *Additional Arguments* instead of *Secondary Arguments*. The **core event extraction** task requires to extract the events with their participants while the **event enrichment** task requires to extract the additional arguments. The **negation and speculation recognition** requires to indicate the negation and/or speculation based on the extracted and enriched events. We denote that though the original definition of *Binding* events can involve more than two arguments, the number of *Binding* events with more than two arguments is small. Most proposed solutions choose to ignore the

Binding events with more than two arguments for simplicity. For arguments, the names before parentheses are types of arguments while the names in parentheses are the types of the target entities allowed be arguments for this event.

According to the **core event extraction** task, the first five events are referred to as single argument events (**SVT**) and the last three events are referred to as regulation events (**REG**).

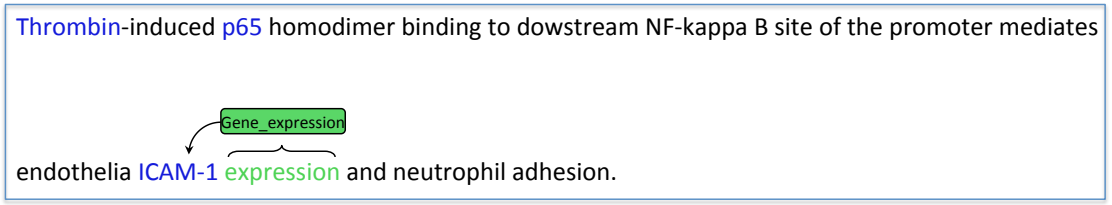
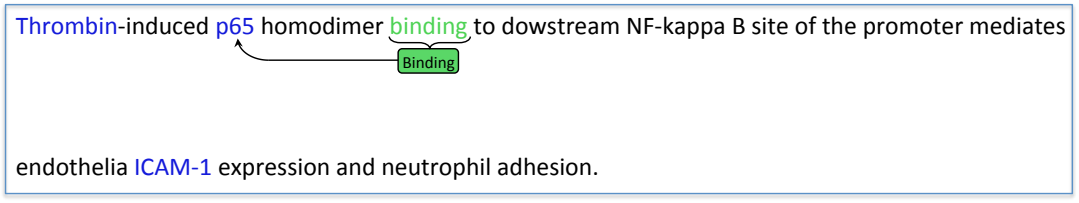


FIGURE 1.1: Example of *Gene_ expression* event.

Figure 1.1 illustrates an example of *Gene_ expression* event: given three protein names ‘‘`Thrombin`’’, ‘‘`p65`’’, ‘‘`ICAM-1`’’ marked in the sentence, the algorithm has to recognize the word ‘‘`expression`’’ is an event trigger of type *Gene_ expression* along with the corresponding argument, which is the protein ‘‘`ICAM-1`’’. This is representative of all the **SVT** events.



(A) Example of *Binding* event with single argument

(B) Example of *Binding* event with two arguments

FIGURE 1.2: Example of *Binding* event.

Figure 1.2 illustrates two examples of *Binding* events: the one in Figure 1.2a is similar to the **SVT** events, whereas the one in Figure 1.2b has two arguments. The second *Binding* event requires to recognize a ternary structure between the trigger and two proteins.

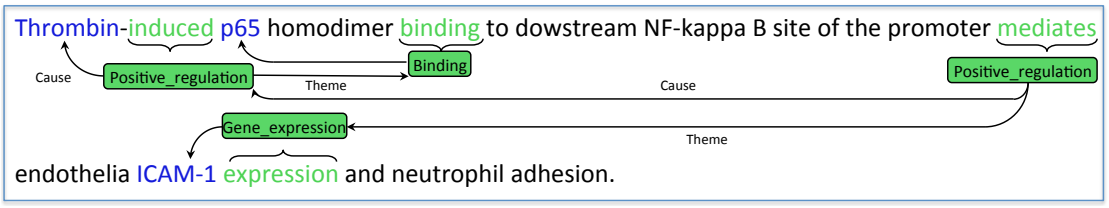


FIGURE 1.3: Example of **REG** events.

Figure 1.3 illustrates two **REG** events along with one *Gene_ expression* event and one *Binding* event. In this example, the extracted *Gene_ expression* and *Binding* events act as arguments of **REG** events. Moreover, **REG** event with trigger ‘‘`induced`’’ is the argument of another **REG** event with trigger ‘‘`mediates`’’. It is obvious that extracting the dependent events before is crucial for extracting the **REG** events presented in the sentence. We call this kind of events **recursive events**. Note that all the **recursive events** are **REG** events, but **REG** events are not necessarily **recursive**. The extraction of the **recursive events** are much more difficult than other events due to the error propagation of event extraction. Hence, **SVT** and *Binding* events are usually measured together since they do not require extracting dependent events in advance. In this thesis, we call the **SVT** and *Binding* events **Non-REG** events. Besides the common difficulties of structured extraction, the BioNLP Genia task merged hierarchical events and parallel events. For example, *Transcription* and *Gene_ expression* are two classes in the task definition while *Transcription* is theoretically a special kind of *Gene_ expression*. The same remark applies to the three **REG** events.

In 2009, the BioNLP task provided the training data by extracting a part of events from the Genia event ontology[1]. Since the event definition/format used in Genia event ontology and the definitions of BioNLP task are not completely consistent, the task organizers introduced partial static relation annotation [14, 15] to extract the events for training and test. The data for the training and development sets was derived from the publicly available event corpus [16] and the data for the test set was derived from an unpublished portion of the corpus. All the data in 2009 were extracted from the abstracts of biomedical articles. In 2011, the task organizers added new documents into the corpus that were extracted from every part of full articles. In 2013, the task organizers provided a new corpus that contains documents from every part of full articles. As the task definitions were modified in 2013, they did not put the data sets of 2011 and 2013 together. But using the previous data sets is encouraged since the new corpus is much smaller and is not sufficient for proper training. Table 1.2 lists the number of articles and documents used in previous tasks, where a document refers to the file that is given by the task organizers, which usually corresponds to a paragraph of an article.

TABLE 1.2: Data sets in BioNLP Genia tasks.

		Training		Development		Test	
		Abs	**Full**	**Abs**	**Full**	**Abs**	**Full**
2009	Articles	800	0	150	0	150	0
	Documents	800	0	150	0	260	0
2011	Articles	800	5	150	5	260	4
	Documents	800	108	150	109	150	87
2013	Articles	0	10	0	10	0	14
	Documents	0	222	0	249	0	305

See Appendix B.2 for more details about the number of words, sentences, events, *etc.*

Though the formulas of task definitions in the three BioNLP Genia tasks are consistent, the corresponding examples in these data sets are not fully coherent. First, some of the annotations provided in the BioNLP 2009 task use the same trigger entity for different

[1] http://www.nactem.ac.uk/genia/genia-corpus/event-corpus

kinds of events, so that trigger detection becomes a multi-label problem. Considering the method that is used to generate the annotation, events may be associated to several categories; hence multi-label classification is required to infer these categories. But in the BioNLP 2011 and the BioNLP 2013, all the newly added trigger entities were labeled as signal event type, where several annotators may however chose different ground truths. To the best of our knowledge, all the participants treated the task as a multi-class problem. Contrary to abstracts, descriptions in full articles contain many coreference problems. As BioNLP 2011 and BioNLP 2013 added full articles, the coreference problem is introduced into the event extraction task. The coreference annotations are given in the training data of BioNLP 2011 and 2013 tasks, but the coreference task is only explicitly integrated in the task definition of BioNLP 2013. Little effort has been down on the coreference problem in this task. Besides, BioNLP2013 corpus contains errors stemming from the processing of the article library, such as missing spaces between words. The task organizers expect the developers to construct a system that can directly work on the original data, hence leave the editing errors as noisy data.

1.2 Thesis Contribution

We now describe our contributions to the BioNLP Genia event extraction task. They can be split into two parts: 1) the problem modelization that contains how we decompose the structured prediction problem and reconstruct the final outputs, 2) investigations about the classifiers and generation of features.

1.2.1 Recursive Pairwise Relation Extraction

We present a new model that is between the pipeline models and joint models that were widely studied in previous works. Our final model is described in Chapter 4 as an extension of a simpler model presented in Chapter 3. These two models constitute the main contributions of this thesis.

Unlike previous models that evaluate the candidate trigger entities and the pair-wise relations separately, our first model predicts (TRIGGER, ARGUMENT) pairs jointly. Starting from pairs enables our model to avoid losing potential triggers, which is the main default of pipeline model, without much additional computational complexity. In addition to the pair-wise predictions, our best model creates the essential part of events through a recursive process. This incremental method constructs the recursive events step-by-step on the most confident predictions, hence does not need any complex inference. Predicting pairs may also lose some potential triggers that could be captured by joint models, which evaluate all the relations altogether via a global score. However, our model slightly outperforms the best joint model on the test set of BioNLP 2011 Genia task. Since the difference between our model and the best joint model can be caused by some implementation details, we say that our model and the best joint model have comparable performance. Note that the training time of our model is much shorter than the training time of the best joint model (30 minutes versus 8 hours and 30 minutes). Our model reaches a good balance between the performance and computational complexity.

1.2.2 Classifier and Feature Design

We tested many classifiers for the multi-class classification problem, F_1-score maximization with unbalanced data. We used asymmetric cost-sensitive SVMs instead of standard SVM to handle these issues, yielding remarkable improvement.

Based on the cost-sensitive binary SVMs, we optimized the macro-average F_1-score by tuning the thresholds along with the output scores computed through cross-validation. Many optimization methods have been tested to maximize the macro-average F_1-score without derivatives. In order to ease the search of thresholds by some methods that are sensitive to the interval of variables, we mapped the output scores of SVMs into $[0, 1]$ by logistic transformation. We tested multinomial logistic regression, binary logistic regression in one-vs-rest framework and soft-max method as logistic transformation. Unfortunately, these optimization solutions do not produce significant differences with our model. The precision-recall curves show that ameliorating classifier selection would not bring significant improvement for our model on this task.

Based on the representation methods used in previous works, we ignored some of the relational features when they cannot be well represented by current encoding methods. Current methods for representing the relations between two entities in a sentence are ineffective when the two entities are far from each other. Abandoning the relational features with respect to the distance between entities improves the classification performance. We also investigated the impact of current encoding methods of relational feature. The results shown that the poor representation of the relational feature may be the bottleneck since the experiment using high-dimensional features shown worse performance. Besides, we applied a language model reported in previous work to learn dense representation of words and integrated the dense features into our model. Our experimental results shown that this dense representations are not good alternatives to words when used in our classification protocol.

1.2.3 Implementation

We implemented our models mainly in Python2.7 along with the natural language processing library (NLTK[2], machine learning library (scikit-learn[3]), scientific mathematics libraries (SciPy[4], NumPy[5]) and statistical library (Statsmodels[6]). Besides, we developed our preprocessing pipeline based on the API provided by Stanford NLP packages [7] in Java. Our code is open accessible on GitHub (https://github.com/XiaoLiuAI/RUPEE).

1.3 Outline of the Thesis

• **Chapter 2** presents the state-of-the-arts approaches that participated in the BioNLP Genia shared tasks. Except the main models used in these approaches, we summarized the preprocessing and feature engineering methods used in these systems.

[2] http://www.nltk.org/
[3] http://scikit-learn.org/
[4] http://www.scipy.org/
[5] http://www.numpy.org/
[6] http://statsmodels.sourceforge.net/
[7] http://www-nlp.stanford.edu/software/index.shtml

• **Chapter 3** explains our first pair-wise model that participated in the BioNLP 2013 Genia task. This model merged the entity recognition step and binary relation assignment step and outperformed the winner of the BioNLP 2013 Genia challenge. In addition, we introduce our tricks regarding feature engineering and preprocessing.

• In **Chapter 4**, we introduce our recursive pair-wise model, that builds on our pair-wise model mentioned in previous chapter and solves the recursive event extraction problem efficiently. This model achieved the best result so far on the BioNLP 2011 and 2013 test sets for a single model with much smaller computational complexity than the joint model.

• In **Chapter 5**, we collect the variants of classifiers, features and models tested during the thesis. One part of these variants were tested to get the optimal solution we presented above, others were developed for further research. However, due to the time limit, we only tried some simple solutions that did not work well.

• **Chapter 6** presents our concluding remarks and explores some further research directions.

The supplements proposed at the end of this thesis are:

• **Appendix A** lists the details of linguistic concepts used in this thesis.

• **Appendix B** describes the details of the BioNLP Genia task that we participated in, including the task definitions and difficulties.

Chapter 2

State-of-the-Art in Biomedical Event Extraction

2.1 Introduction

In this thesis, we address some tasks of event extraction from biomedical text proposed in BioNLP shared tasks [1–3]. The BioNLP event extraction tasks aim to extract molecular interactions mentioned in biomedical text and organize them into structured format so that dedicated biomedical databases can be automatically constructed. This chapter introduces the state-of-the-art biomedical event extraction approaches mostly as reported in previous BioNLP workshops. Most current NLP approaches that try to understand language decompose human language into basic morphemes and then reconstruct high-level information step by step. Hence, most NLP systems designed to solve complex tasks first rely on simpler systems that perform simple, yet crucial preprocessing of text. Event extraction is such a high-level NLP task that requires the support from many low level NLP tasks such as tokenization, sentence splitting, Part-of-Speech tagging, syntactic parsing, *etc.* In this chapter, we summarize the frequently used methods and features for every step of existing systems starting from scratch. Besides, we also explain how previous models formulate the complex event extraction problem along with the extracting algorithms they used.

Section 2.2 describes the preprocessing approaches used to analyze the text at different levels. Section 2.3 introduces the popular features extracted from the outcomes of preprocessing steps. Section 2.4 summarizes the proposed approaches aimed to solve BioNLP Genia event extraction tasks.

2.2 Text Preprocessing

Preprocessing text for event extraction is a big pipeline process that involves many other NLP applications, where the outcomes of preceding steps are used as input for succeeding steps. Consequently, any change in the preprocessing sequence can make the final output different. Besides, every application in this pipeline is trained on a special data-set which can be different from the target set of the final task. This can be problematic especially for some high-level tasks. For example, the labeled training set used to train syntactic

parsers is set up using the result of special tokenization and sentence splitting approaches. Hence, to build the most effective event extraction methods, one must take care of using the best performing combination of preprocessing approaches. We present in the rest of this section our conclusions with a focus on tokenization and parsing. Since our major goal is to develop biomedical event extraction models, we did not develop original preprocessing strategies. We list here all the methods used in reported approaches along with their performances.

2.2.1 Tokenization and Sentence Splitting

Tokenization is the process of breaking a stream of text up into words, phrases, symbols, or other meaningful elements called tokens. Those tokens are used as input for further processing such as tagging, chunking, parsing, *etc.* In English, words are separated by white-spaces and punctuations, so that a simple heuristic can handle most situations by defining tokens as continuous alphabetic characters or numbers. Apart from the general case, contractions like "can't" or hyphenated words like "New York-based" cannot be well processed by such simple heuristics. It is similar for sentence splitting. Sentences are usually split by some special symbols like periods, interrogations and exclamatory marks. Some exceptions can be the periods between names like "A. Bordes", between special terms such as "Dr.", "U.S.", and in numbers ".02%". In general, tokenization and sentence splitting is not very complex, thus most tokenization and sentence splitting applications are rule based approaches. However, when processing texts from specific domains, efforts from experts are usually required to improve these rule based approaches to handle such exceptions.

2.2.1.1 Difficulties in Biomedical Documents

As described above, even for languages that use inter-word spaces (such as most that use Latin alphabet), there are many exceptions that cannot be well solved by heuristics like regular pattern expressions. This problem becomes even more serious in biomedical texts because of the biomedical or chemical formulas and jargons. Strings like "CD19+CD27-", "30-38 kDa", "[6,7]", "p<=0", "PKD1/3-/-" are very confusing not only for tokenizers but for other NLP applications like tagging or syntactic parsing. Dealing with compound words is crucial for good performance since they are used very frequently in biomedical documents.

Tokenization Hyphenated words and parallel items separated by slashes are the most problematic cases for tokenization. Hyphenated words can be divided into three categories: first, compound words that contain only normal words like "immunobead-isolated", "dose-dependent"; second, complex biomedical terms like "BMP-RII-Fc", "BMP-RIB"; third, mix of biomedical terms and normal words/prefix/suffix like "anti-IgM", "IRF-4-expression". We define "normal words" as the non-hyphenated words that are not biomedical acronyms or identities of chemical materials. Except pure biomedical terms, the first and third categories of words need special attention. In training examples, some event triggers are just parts of hyphenated words such as "expression" in "IRF-4-expression". Recognizing these triggers is very difficult if one does not split them during the tokenization. Besides, compound words increase the sparsity of feature vectors because current encoding methods cannot represent the relations between compound words

and their sub-words. However, not all the hyphenated words are suitable for breaking including some commonly used words and some biomedical terminologies. Simply splitting all the hyphenated words can make ambiguous tokens and change the sentence structure. Separating parallel items by slashes creates difficulties both for low-level processing and high-level understanding. During processing, it is not easy to distinguish whether slashes are used to separate parallel items or special symbols like fraction signs or other symbols in biomedical articles. For syntactic processing, the necessity of splitting parallel items is also arguable when people omit some common parts of items. For example, one can write "IL-1/2/3" or "IL-1/-2/-3" instead of "IL-1/IL-2/IL-3" for short. Moreover, slashes are ambiguously used as "and"/"or" in different context.

Sentence Splitting Sentence splitting of biomedical text is easier than tokenization thanks to good typesettings, which strictly use new line or periods followed by white-space at the end of sentences. The only mis-broken sentences are those containing authors' names in references. Look at the example sentence below.
Previously we have shown that PKDs have an essential role in regulating class II histone deacetylases in DT40 B-cells [Matthews, S.A., Liu, P., Spitaler, M., Olson, E.N., McKinsey, T.A., Cantrell, D.A. and Scharenberg, A.M. (2006) Essential role for protein kinase D family kinases in the regulation of class II histone deacetylases in B lymphocytes. Mol. Cell Biol. 26, 1569-1577].
Many periods exist in the names of authors and volume information and the entire titles of cited articles, which are also sentences, are included in references. This is an extreme example that, no matter whether this sentence is correctly split or not, will confuse a syntactic parser. Generally, references in our target corpus insert the author's name, abbreviation of department, addresses, *etc.* at the end of sentences. Besides, some abbreviations of domain specific terms also contain periods like the author's name, such as *E. Coli*, which refers to the bacteria *escherichia coli*. Breaking sentence at those kinds of periods should not hurt BioNLP event extraction models since there is no example of event in references appearing in the training set.

2.2.1.2 Implementations

This section summarizes the tokenizations commonly used in BioNLP event extraction tasks: support tokenization, which is provided by the task organizers; Stanford tokenization, which is generated by the Stanford tokenizers specially developed for this task; GDep tokenization, which is integrated in the Genia Dependency parser (GDep) parser.

• Support tokenization is the resource provided by the task organizers, which aims to prevent the participants from putting too much efforts on the low-level tasks. It is generated by the script GTB-tokenize.pl[1] that attempts to mimic the tokenization used by the Genia Treebank. Genia Treebank[2] is a biomedical corpus used for syntactic parsing tasks that contains 200 documents and 300 abstracts. This tokenization preserves the hyphenated words, making the sentences more compact and easier to read by humans, because humans can easily decode the inner structure of compound words or parallel items concatenated by slashes, and a shorter sentence is always clearer and easier to

[1] https://github.com/ninjin/bionlp_st_2011_supporting/blob/master/tools/GTB-tokenize.pl

[2] http://www.nactem.ac.uk/tsujii/GENIA/topics/Corpus/GTB.html

understand. That is also easier to parse by syntactic parsers when hyphens and slashes are not broken as syntactic parsers not aware of the semantic meanings of words. Thus, support tokenization does not apply any task-specific word breaking strategy, so that hyphens and slashes are never separated.

• Stanford tokenizer is developed by Mcclosky as an integrated part in his BioNLP parser [17]. This tokenizer is specially designed to handle the BioNLP event extraction tasks. Thus, it breaks specified hyphenated words by handcrafted heuristics that use specified suffixes and prefixes such as "-dependent", "-defective", "-negative", "anti-", *etc.* Besides, all the words connected by slashes are separated except some predefined exceptions like "-/-". To the best of our knowledge, this tokenizer was used by UCLEED, TEES and TEES 2.1 [18–25] systems. The success of this tokenization is mainly because splitting into fine-grained tokens increases the chances of recognizing potential triggers and relations between parallel items concatenated by slashes. However, this tokenization cannot handle new compound words, which are not in the predefined lexicon. It also has some drawbacks. Consider the tokenized sentence below:

Each bar represents mean+ / - SEM of at least three independent experiments.

In this sentence, "mean+/-SEM" is split into four tokens "mean+", "/", "-", "SEM" by this tokenizer.

• GDep is a dependency parser for biomedical text developed by Kenji, which is freely available from people.ict.usc.edu/∼sagae/parser/gdep/. This parser contains a simple tokenizer that splits tokens when encountering a set of specified symbols and words. As the Genia Treebank tokenizer, it does not contain any specific solution for hyphenated words, slashes or biomedical jargons. Unlike support data and Stanford parser, which contain both tokenization and sentence splitting solutions, GDep parser requires to split sentences beforehand.

Tokenization is used to extract the minimum semantic units for many NLP tasks. However, different tasks have their specific requirements for the minimum semantic units.

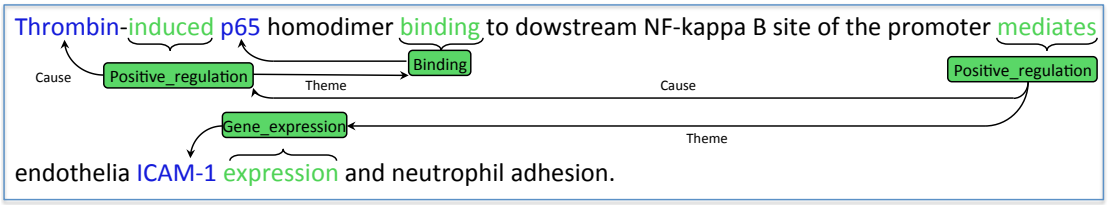


FIGURE 2.1: Trigger and argument in one token.

The event trigger **"induced"** and its argument **"Thrombin"** displayed in Figure 2.1 are different parts of a noun phrase **"Thrombin-induced"**, which means that the minimal semantic unit considered for BioNLP Genia event extraction should be smaller. Support tokenization and GDep do not apply any task specific tokenization strategy, thus are less powerful in this task. Stanford tokenization, which applies some task specific tokenization rules, was used by the best models UCLEED, TEES, *etc.* Hence, we believe that Stanford tokenization is the best among other solutions. Moreover, considering the drawbacks of the Stanford tokenizer, a better tokenizer should integrate domain knowledge to be more precise.

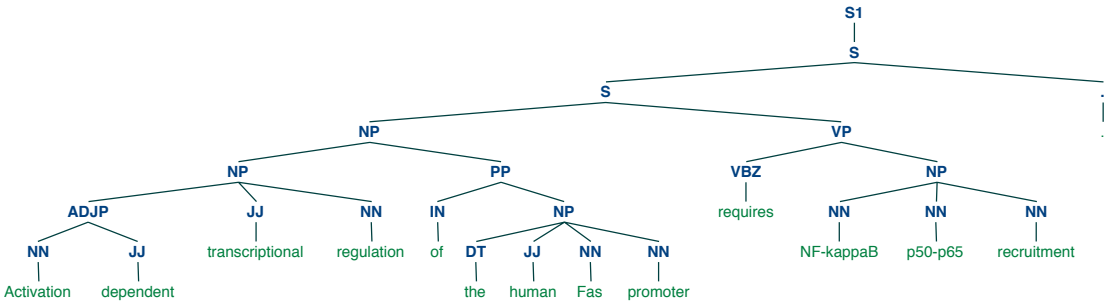


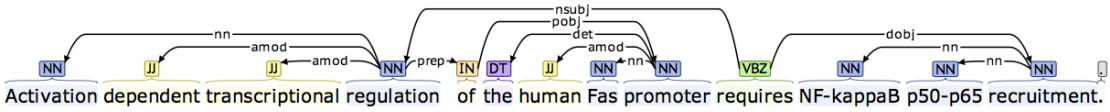
FIGURE 2.2: **Example** of phrase structure tree in Penn Treebank format.

2.2.2 Syntactic Grammars

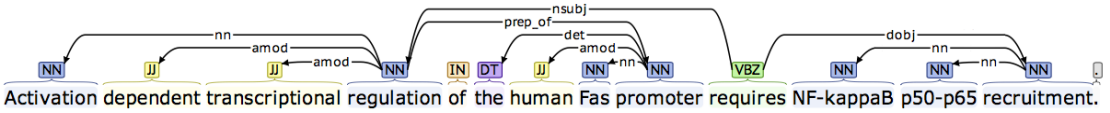
Human languages organize sentences following some common paradigms, which are referred to by linguists as grammars. In computational linguistics, these grammars help computers to organize the sequence of tokens in a sentence into more informative structures. Two kinds of widely used grammars are phrase structure grammars and dependency grammars: phrase structure grammars view the sentence in terms of the constituency relations, whereas dependency grammars represent sentence in terms of predicates and their arguments (the dependency relations). These grammars are imperfect and there are always exceptions that cannot be well handled by them in practical NLP tasks. However, they provide features which are essential to all state-of-the-art systems for biomedical event extraction.

2.2.2.1 Phrase Structure Grammar

In linguistics, phrase structure grammars are based on the constituency relations. Hence, phrase structure grammars are also known as *constituency grammars*. The constituency relation derives from the subject-predicate division of grammars, of which basic clause structure is subject (noun phrase NP) and predicate (verb phrase VP). A widely used format of phrase structure grammar is the Penn Treebank (PTB) format, which defined series of part-of-speech for both tokens and sub-phrases. Note that part-of-speech (POS) is a linguistic category of words, which is generally defined by the syntactic or morphological behaviors of the lexical item in sentence, such as noun, verb, adjective. Table A.1 in appendix lists all the POS tags used in PTB project. Figure 2.2 presents an example of phrase structure tree generated by Mcclosky's biomedical syntactic parser for sentence *Activation dependent transcriptional regulation of the human Fas promoter requires NF-kappaB p50-p65 recruitment.* In phrase structure grammar, each token is connected to one or more nodes in the grammar tree. This kind of grammars is usually not directly used in BioNLP event extraction tasks, because it does not directly describe the relations between tokens. However, Stanford NLP package provides methods to convert the phrase structure grammar into dependency grammar and some of dependency grammars were generated based on the phrase structure grammar.



(A) Basic dependencies



(B) Collapsed and propagated dependencies

FIGURE 2.3: **Stanford** dependency parse graphs.

2.2.2.2 Dependency Grammar

Unlike phrase structure grammars, that concentrate on subject-predicate relations, dependency grammars focus on predicate-argument relations. There is a one-to-one correspondence: each token in a sentence corresponds to just one node in the syntactic structure. Many methods are proposed to represent dependencies, which can be trees, directed acyclic graphs or other formats. Figure 2.3a illustrates a visualization of dependency graphs in the Stanford dependency format that correspond to the example sentence of phrase structure of previous section. The dependency grammars are usually used in event extraction because they directly describe the relations between tokens. However, since it can be redundant to describe all relations with prepositions like "of", Stanford parser proposed a way to collapse and propagate the dependencies as shown in Figure 2.3b to make the dependency graph more compact. Unfortunately, dependency grammars cannot handle relation between ``activation'' and ``dependent'' appropirately. In this sentence, ``regulation'' is modified by two independent modifiers: ``activation dependent'' and ``transcriptional''. But the dependency path between ``activation'' and ``dependent'' is the same than the path between ``activation'' and ``transcriptional'', which means one cannot realize whether ``activation'' and ``dependent'' form an adjective phrase.

As described above, there are many representation methods of dependencies. We introduce two widely used formats: CoNLL format proposed by the Conference of Natural Language Learning 2006, and Stanford format proposed by Stanford NLP Group. For every token, CoNLL format specifies 10 attributes listed in Appendix Table A.2, which index each token and specify its POS and index of its dependency head. It is the most widely used dependency format and represents a dependency tree, where each token can only have one head. Unlike CoNLL format, Stanford format lists pairwise relations between indexed tokens in every line. This format can be used to represent any graph, so that one token can have multiple heads. Unlike CoNLL format, Stanford format does not give the POS of tokens. This format is used in projects supported by Stanford NLP group, and is the most frequently used format in applications that participated in BioNLP 2011 event extraction tasks [26].

2.2.3 Frequently Used Parsers

In this section, we describe the parsers used in previous works that participated in BioNLP Genia tasks.

2.2.3.1 Stanford Series

Stanford parsers provide both phrase structure tree and dependency graph. Two versions were used in previous BioNLP challenges: standard Stanford parser and biomedical specified parser provided by McClosky. The Standard Stanford parser is trained on general text and provides both phrase structure in PTB format and dependency graph in Stanford dependency format. McClosky-Charniak parser is trained on Genia Treebank corpus and provides only phrase structure in PTB format. McClosky-Charniak parser is widely used in many approaches (10 out of 16 participants in BioNLP 2011 Genia task) including the best pipeline model TEES and the best joint model UCLEED. However, they always used this parser with a task specified tokenization, which is different from the tokenization contained in the Genia Treebank. McClosky claimed [3] that this combination of tokenization and parser gave better performance in BioNLP event extraction tasks. However, we cannot assess the quality of parsing itself on BioNLP data, since no labeled data for parsing is provided for evaluation. The better performance in event extraction cannot be directly connected to the parser quality. We conjecture that the better performance can be caused by a fine-grained tokenization bringing essential information for extracting triggers in hyphenated words.

2.2.3.2 Other Parsers

Apart from Stanford related parsers, many other syntactic parsers were used in previous work. We list below the support resources provided in BioNLP tasks. Most of them were trained on the Genia Treebank corpus.

- Enju parser [27] provides both phrase structures and dependency parsing results. This parser defines its own specified PTB-like format and predicate-argument structure. It also gives CoNLL format outputs.

- GDep parser [28] does not provide phrase structures but dependency parse trees in CoNLL format.

- CCG parser provides phrase structures in PTB format and dependency parse graphs in Stanford format.

Note that Enju parser returns very good parsing results, [29, 30] used it along with the McClosky-Charniak parser while Kim et al. [31] used only Enju parser. Berkeley parser, which was trained on general text and only provides phrase structures in PTB format, was not used in any participant's system to the best of our knowledge.

2.3 Dedicated Feature Engineering

Based on our choice of preprocessing, we developed many features. Unlike acoustic or visual data, texts and parsing results are symbolic but not numeric. A common way of

[3] This was appearing on the web link `http://nlp.stanford.edu/software/eventparser.shtml`. Disappeared in or before June 2014

TABLE 2.1: Features used by our system

Candidate entity features	Base form (stem) of the head token.
	Base form of the head token without '-' or '/' before of after.
	Sub-string after '-' in the head token.
	POS of the head token.
	First token of the entity is after '-' or '/'.
	Last token of the entity is before '-' or '/'.
	Head token has a special prefix: "over", "up", "down", "co"
	Concat. of base form and POS of parents of the head token in dependency parse.
	Concat. of base form and POS of children of the head token in dependency parse.
	Base forms of k neighboring tokens around the entity.
	POS of k neighboring tokens around the entity.
	Neighborhood of the entity has '-' or '/'.
	Sentence has "mRNA".
	Entity is connected with another string using support tokenization.
Argument features	POS of the head token.
	Features extracted from IntAct when the argument is a protein.
	Base forms of k neighboring tokens around the argument.
	POS of k neighboring tokens around the argument.
	Concat. of base form and POS of children of the head token in dependency parse.
Joint features	Token sequence between candidate and argument has proteins.
	V-walk features between candidate and argument with base forms.
	E-walk features between candidate and argument with base forms.
	V-walk features between candidate and argument with POS.
	E-walk features between candidate and argument with POS.
	Candidate and the argument share a token using support tokenization.
	Feature extracted from IntAct when two arguments are both proteins.

representing symbols is by encoding them by dictionaries, where each symbol is represented by its index in a dictionary. Since indices are independent of the significations of the symbols, current models do not use any semantics underlying the symbolic system. Consequently, most of NLP applications require handcrafted heuristics to acquire useful features. In this section, we introduce the most commonly used features in event extraction. To be consistent with our models presented in the following chapters, we arrange these features into three groups: unitary features, pairwise features and global features. As illustration of features one can use for biomedical event extraction, we list in Table 2.1 the actual features we used in our system.

2.3.1 Unitary Features

The target entry of an event is not necessarily a single token, especially under different tokenizations. Hence, we define an entity as a continuous character chain, which can be a part of a token, a token, multiple tokens, such as "anti-" in "anti-IgM", "comparable level". Unitary features are created to describe the properties of target entities. There are two categories of unitary features: features of target entities and features of context.

Many of these features are based on outcomes of preprocessing steps, which correspond to a certain tokenization. When an entity covers more than one token, most approaches implement features based on the most representative token of the entity. These tokens are found by handcrafted heuristics and are called *head tokens*. For short entities that are usually a part of token, someone used fine-grained tokenizations to avoid the existence of small entities inside words. Besides, some morphological features were developed to describe these small entities when they are not isolated even by fine-grained tokenizations.

2.3.1.1 Features of Target Entities

Target entities are described using morphology and POS features. Due to the morphological changes in English, one word can have many variations such as "regulated" and "regulates", which are not interconnected by models that use dictionaries directly. In addition, compound words like "upregulated" cannot be associated with neither "up" nor "regulate". Thus, most methods use stemmed words, lemma of words, prefixes, suffixes of words as features, hoping that models can discover the associations between related words this way.

The stem of a word is the part of the word that is common to all its inflected variants. For example, "standard" is the stem of "standardize", "stabil-" is the stem of "destabilize". However, current word stemmers only deal with the suffixes but not the prefixes of words. They apply a set of morphological rules to recognize and cut specified suffixes of words.

The lemma of a word is the common dictionary form of a set of words. The difference between a lemma and a stem is that a lemma is a word itself while stem is the root form, which is not necessarily a word. Given two words "regulations" and "regulated", they share the same stem "regul", but have different lemmas "regulation" and "regulate". It is clear that stem and lemma map different forms of words into a general term to increase the generalizability. However, neither of them deal with the prefix and sometimes the discarded suffixes can also provide useful information.

To overcome the weakness of stems and/or lemmas, the first and last n characters are also usually used to describe the prefix and suffix, where n is specified by the system. Another way of describing suffixes or prefixes is to detect some specified forms such as "up", "down", "over", "co", *etc.* In biomedical text, hyphens are very important as a specific prefix or suffix in the cases like "positive- or negative- A", "B -regulation".

Compared to morphologic features, POS features are much simple, since the POS of the head tokens for target entities is simply used.

2.3.1.2 Features of Context

A word or an entity may have different significations under different contexts. It is impossible for models to make correct predictions without context information. A straightforward method is to take the neighboring tokens of target entities into account. Two methods were proposed by using different definitions of neighboring tokens.

Word Window Given a sentence, which is a sequence of tokens, neighboring tokens can be naturally defined as the tokens before or after the target entities (or the head tokens of entities). Involving neighboring tokens is actually observing the local context centered around the target entity with a window. The number of neighboring tokens is called window size. Though all the unitary features based on tokens can be used for the neighboring tokens, stemmed word and POS are the most commonly used. Due to the flexibility of natural language, the informative neighboring tokens are not always in the window range or at the same relative position.

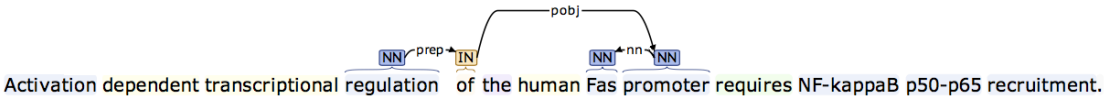
Dependency Adjacent Nodes Given a target entity, which is represented by a head token, one can find its adjacent nodes in the dependency tree/graph. A target entity can be a candidate trigger entity or protein entity in the BioNLP event extraction tasks. Since dependency grammars encode the direct interactions between tokens, heads and modifiers of head token provide crucial information for disambiguation of neutral words. These dependency relations are encoded in a bag of pairs. For the token "regulation" in Figure 2.3b, {(nn, Activation), (amod, dependent), (amod, transcriptional), (prep_of, promoter), (requires, nsubj)} are used as features. Tokens in those pairs are usually replaced by the stem of words or POS. Looking the adjacent nodes as parents and children nodes, one can also add ancestors and descendants with a specified depth as features.

2.3.2 Pairwise Features

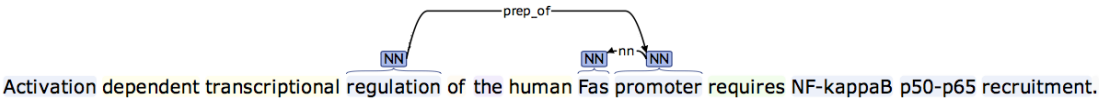
In structured information extraction, a pairwise relation is the basic relation because more complex structures can be represented by sets of pairs. Usual approaches either construct complex structures either by greedily aggregating pairs step by step or by searching the structure that maximizes a global score defined as sum of unitary elements and pairwise relations. Pairwise features are the crucial features in all the structured information extraction tasks.

2.3.2.1 Dependency Path

Most pairwise features are based on the shortest dependency path. For two tokens, the shortest dependency path is the shortest path between corresponding nodes in a dependency parse tree/graph. In the rest of this thesis, we call it ***dependency path*** for short. Figure 2.4 illustrates the dependency path between the trigger word "regulation" and the protein "Fas". It is clear that collapsed and propagated dependency graphs provide shorter dependency paths that are easier to understand. Thus, collapsed and propagated dependency graphs are used in almost all the approaches that used Stanford related parsers. Besides, trimming is a very useful technique used in previous works including both symbolic approaches by Buyko et al. [32] and statistical ones by Riedel and McCallum [20]. It is usually a task specific solution that removes some pointless edges using handcrafted rules to generate shorter essential dependency paths. These two solutions play a similar role in NLP applications, that is to produce shorter dependency paths for high-level processing, and are frequently used together. We noticed that, the dependency graph of a sentence is not necessarily a connected graph, because, sometimes, the dependency path between two tokens does not exist.



(A) Shortest dependency path in basic dependencies



(B) Shortest dependency path in collapsed and propagated dependencies

FIGURE 2.4: **Shortest** dependency path

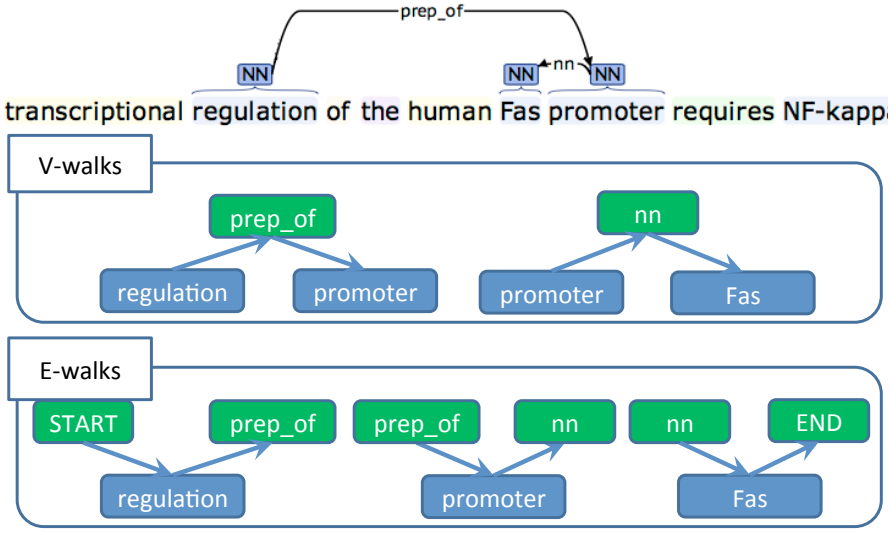


FIGURE 2.5: ***E-walks*** and ***V-walks*** features derived from a dependency path.

2.3.2.2 Encoding Methods

A dependency path is a variable length sequence, which cannot be directly used as features in usual NLP models. Following the N-grams encoding methods, which is widely used in NLP applications, previous methods represent the dependency paths by bags of N-grams in the sequence. We summarize the encoding methods into two categories: the first methods represent the dependency paths by trigrams including both dependency tags and tokens; the second methods represent the dependency path by n-grams involving either only tokens or dependency tags.

E/V-walk In the first kind of methods, the dependency path is treated as a sequence constituted by staggered edge nodes and vertex nodes. This sequence can be represented by trigrams of nodes. These trigrams can be defined in two ways: trigrams containing two vertex nodes are called *V-walk*, whereas trigrams containing two edge nodes are called *E-walk*. Figure 2.5 shows the *E-walk* and *V-walk* for the dependency path mentioned earlier. The *START* and *END* special edge nodes are used when there is only one edge in the dependency path. Those triplets are further encoded into tuples (*dep-tag*, *token*, *dep-tag*) for E-walk; and (*token*, *dep-tag*, *token*) for V-walk. After replacing the

tokens by stemmed words or POS, those triplets are transformed into joint symbols "*dep-tag-stem-dep-tag*", "*dep-tag-POS-dep-tag*" for E-walk; and "*stem-dep-tag-stem*", "*POS-dep-tag-POS*" for V-walk. In BioNLP tasks, the tokens are usually replaced by an indicator *PROT* when it is annotated as part of a protein like "promot->-NN->-PROT", where the arrows indicate the direction of dependency edges. In order to increase the generalizability of features, the UCLEED model [20] creates new features by masking one of the three elements in trigrams by a specified symbol. For example, "{*regul-**-promot*, *regul->-PREP_of->-***, ***->-PREP_of->-promot*}" can be created from "*regul->-PREP_of->-promot*".

N-Grams The second kind of methods only considers one type of nodes in dependency path and formats them into bag of n-grams, where n is a hyper-parameter of the system. In order to avoid duplication with *E/V-walk*, n is usually larger than 2. For example, "*START->-PREP_OF->-NN*" is a trigram of dependency edge nodes. Similar to the *E/V-walk*, tokens are replaced by the POS or stem of words and proteins are replaced by a specified indicator in the n-grams of vertex nodes.

2.3.3 Other Features

All the features listed above are domain independent, and can be used in any type of structured information extraction tasks. Even if participants may slightly modify the implementation details for special tasks, these features basically follow standard principles. Apart from these features, some highly task dependent features also are implemented. For example, one can add an indicator of the presence of key word "mRNA" in the sentence to help distinguishing the *Transcription* from the *Gene_expression*. This feature actually improves the performance, but is not logically reasonable because the key word "mRNA" does not necessarily modify the candidate trigger. We used the indicator of "mRNA" in our systems.

2.4 Previous Work

In this section, we introduce the approaches developed to solve the BioNLP Genia tasks in BioNLP challenges 2009, 2011 and 2013. We divide them into three categories: pipeline models, joint models and pairwise models. Since the events can be naturally represented by graphs, previous works extract the events as predicting nodes and edges. The main difference between them is the order of extraction. Pipeline models follow a process of nodes detection, edges detection and graph construction. Joint models solve all together and pairwise models consist in edges detection and graph construction. All the models only solve events where triggers and arguments are in the same sentence since cross-sentence events are too hard to extract and appear rarely. Björne et al. [23] report that only 4.8% of events in the gold annotation of BioNLP 2009 Genia task cross sentence boundaries.

2.4.1 Pipeline Models

Pipeline models build complex events step by step, where each step solves a sub-problem based on the outcome of preceding ones. During the creation of complex events, the outcome of preceding steps can be either used or discarded by succeeding steps. Besides, succeeding steps always observe more global information than preceding ones. Thus, a pipeline can be seen as a process, where each step builds more complex structures and narrows the sample range with richer information under more constraints. It is the largest category of approaches for BioNLP and has many variations in past studies. We introduce this category of approaches in two times: general architecture and implementation.

2.4.1.1 General Architecture

Pipeline models generally follow a general architecture that contains three steps:

1. Trigger Detection. This step extracts the triggers with special event types regardless of their potential arguments or other constraints. In some implementations [33–37], this step can be refined into two sub-steps: candidate entity generation and trigger type assignment, where the first sub-step filters the irrelevant tokens to decrease the sample size.

2. Edge Detection. Given extracted triggers, one can construct the edges between these triggers and their arguments (proteins). Extracted triggers with no arguments predicted in this step are removed but other constraints are ignored such as the fact that triggers can only be arguments when they correspond to a valid event.

3. Post-processing. Based on the extracted pairs, post-processing constructs multi-argument events using either a rule based method or classifiers such as SVM. Edges are removed when they violate any constraint defined by the tasks.

TEES systems [23–25, 38] are typical examples of pipeline models that follow this architecture. However, there are some pipeline models that slightly differ from this architecture. A notable exception is Li et al. [39] that added a second trigger detection and edge detection after the first edge detection step.

2.4.1.2 Diverse Implementations

Many approaches have been proposed as pipeline models, using different methods including rule-based method and statistical learning at each step. We describe some typical methods used in previous approaches below.

Trigger Detection The trigger detection step aims at extracting triggers with their corresponding types. In task definition, triggers are entities that can contain multiple tokens, and named entity recognition is a natural choice for solving this problem such as in [40–42]. Some other works [23, 24, 34, 43, 44] only considered single-token triggers because more than 90% of triggers contain only one token. But this kind of work usually wisely chooses the head token instead of all the tokens in entity during training to reduce

the noise and fuse the consecutive tokens into entities using some handcrafted rules. As one of the official evaluation method of BioNLP, approximate span, counts as correct predictions that match only a part of a trigger, correctly matching a token is enough to tackle the problem in this case.

The trigger detection step may contain two sub-steps depending on the implementation. This kind of models adds a candidate trigger generation step before assigning trigger types to tokens in order to decrease the size of the set of candidate examples. In this case, the candidate trigger generation step usually contains some simple filtering methods whereas trigger type assignment contains a features based multi-class classifier (SVM, *etc.*). Vlachos and Craven [35] and R. McGrath et al. [36] filter the tokens with regard to the POS, where only nouns, verbs, adjectives, adverbs and prepositions are taken into account. Emadzadeh et al. [37] select the tokens matched by a dictionary generated given the gold annotations of training data. Lee et al. [33] and Kilicoglu and Bergler [34] applied both dictionary and POS in their models along with some handcrafted rules. Note that dictionary based methods always contain some morphological solutions like stemming, part string match to improve generalization.

Unlike two-step methods, which differ on filtering strategies, one step methods differ on classification strategies. We classify the methods into dictionary-based methods, rules-based methods, named entity recognition methods, multi-class classification methods and methods that are mixture of them. Triggers are detected with dictionaries created on training data in [32, 44–47], where trigger types are disambiguated by the frequencies. Buyko et al. [32] measure the importance of a trigger t_i for event type T by $Imp = \frac{f(t_i^T)}{\sum_{i'} f(t_{i'}^T)}$, where $f(t_i^T)$ refers to the frequency of trigger t_i appears in type T. Bui et al. [47] measure the importance of a trigger in similar way but replacing the total frequency of all the triggers in type T by the total frequency of trigger t_i in training set. In addition, they also filter tokens with respect to the POS, which only involve nouns, verbs and adjectives. Le Minh et al. [43] combine rule-based and dictionary-based approaches by handcrafted patterns, for instance, NN/NNS + of + PROTEIN, VBN + PROTEIN, where NN/NNS and VBN are POS tags. All the POS tags can be found in Table A.1. All the approaches described above for one step methods only consider single-token triggers; multi-token triggers are solved by named entity recognition approaches such as conditional random field (CRF), maximum entropy Markov model in [40–42]. Other works [23, 24, 30] use multi-class classifiers (SVM, maximum entropy model) to classify the tokens into a valid trigger class or a *None* class. Moreover, MacKinlay et al. [41] merge the results of dictionary-based method and a CRF model.

Edge Detection Even if edge detection uses the outcome of trigger detection as input, there are not as much variations as for trigger detection. Edges are detected by handcrafted rules [40, 41, 45, 48] or by models learned from training set such as [19, 23]. In BioNLP tasks, all approaches using handcrafted rules perform poorly on the official test. Learning models can be divided into three categories: feature-based classifiers, rule learning system and mixture. The BioSEM [47] system, which achieves the best performance on *Binding* event prediction in BioNLP 2013 Genia task, is the most successful rule-based system that we known of in BioNLP Genia task. It learns the rules in predefined patterns by observing some features from a training data-set: the distances and prepositions between entities, POS of trigger, pattern related frequency features. One advantage of this model is that it detects both pairwise edges and triplets for multi-argument events on extracted triggers. Jointly extracting multi-argument events is

probably the main reason of its good performance on *Binding* events. We recall that the *Binding* events mostly involve two argument proteins. In addition, this model does not need any post-processing step. Apart from the rule-based models, most previous works apply a multi-class classifier to solve the edge detection problem. Two types of classifiers are often used: SVMs and maximum entropy models. We note that Buyko et al. [32] tried both maximum entropy with common features and SVMs with graph kernel.

Besides the classification models, previous studies are also different on the way they treat different classes.

Edges are extracted recursively in [34, 35, 42] to solve the recursive events: they enforce the constraint that triggers acting as arguments must have an argument. Other methods predict all the possible pairs regardless of this constraint.

Post-Processing The post-processing completes the event construction based on the pairs extracted by preceding steps. Since the edges of single-argument events directly solve the extraction problem, post-processing step actually deals with the recursive events and multi-argument events. Given extracted pairs, constructing recursive events only requires a simple rule-based approach. The real difficulty is creating multi-argument events. Buyko et al. [32] simply create all the possible combinations that do not violate a set of predefined constraints. For example, given a *Regulation* trigger with two THEME arguments and two CAUSE arguments, they will create four events corresponding to all combinations between THEME and CAUSE. MacKinlay et al. [41] apply similar strategies along with selected thresholds to eliminate events, when distances between two arguments are too long. Another way is to use a classifier trained by machine learning methods, usually an SVM [23, 24, 49], to determine the validity of merging two arguments into an event. Miwa et al. [50] considered both edges inside and outside of the events under the machine learning framework.

2.4.2 Joint Models

Joint models solve the structured information extraction in one step. As the sentences and events can be both represented by graphs, joint models solve the problem by graph matching or inference algorithms that search for event graphs that globally correspond best to input sentences.

2.4.2.1 Markov Random Fields

Three models [18–20] are proposed to solve the BioNLP Genia task by inferring the best event graphs from sentences. Considering a sentence as a graph where the tokens are nodes, extracting events requires searching the best states for each node and for the most probable connections between them using a global score defined by a Markov Random Field. As the number of possible graphs for a long sentence is very large, inference algorithms try to find the target graph by propagating the states of nodes and edges under constraints.

The UCLEED model [20], which won the BioNLP 2011 challenge, is the best joint model reported so far. In order to decrease the computational cost, this model uses a dictionary

to generate a set of potential candidate tokens that will be observed in a sentence. A dictionary is generated from gold annotations, in which one stored sequences of tokens encoding entities together with their sub-tokens. They also stem the last token in entities and manually eliminate some stop words. UCLEED model represents a sentence by a graph $\mathbf{G} = \{\mathbf{e}_i, \mathbf{a}_{i,j}\}$ where $\mathbf{e}_i$ describes the state of token i and $\mathbf{a}_{i,j}$ describes the state of edge between tokens i and j. The appropriateness of the graph can be measured by a score $s(\mathbf{e}, \mathbf{a}) = \sum_{e_{i,t}=1} S_T(i,t) + \sum_{e_{i,j,r}=1} S_R(i,j,r) + \sum_{b_{p,q}=1} S_B(p,q)$ where $S_T(i,t)$ is a per-trigger scoring function that measures how well the event label t fits to token i, $S_R(i,j,r)$ measures the compatibility of role r as label for the edge $i \rightarrow j$, $S_B(p,q)$ measures the compatibility of the protein pairs involved in multi-argument *Binding* events. For recursive events, where events act as arguments of other events, the scoring function is modified with constraints that form an Integer Linear Program, which is solved by dual decomposition.

2.4.2.2 Pattern Matching

Another category of joint models is based on pattern matching that try to recognize a whole event as a symbolic pattern. Hakenberg et al. [51] format the preprocessed texts into database queries and retrieve the single-argument events by subsequently querying the database. MacKinlay et al. [52] and Liu et al. [53] extract events by matching subgraphs on the dependency parse graph. Compared to the Markov Random Field model, pattern matching models do not achieve a remarkable performance.

2.4.3 Pairwise Models

As a trade-off between pipeline and joint models, [54–56] extract the (TRIGGER,ARGUMENT) pairs directly. They can be seen as special pipeline models that choose a different starting point, that is, edge detection instead of trigger detection. TEES and UCLEED models listed in previous two sections are the best pipeline and joint models. UCLEED model has better performance but a larger computational cost than TEES, which shown that considering globally is more complex but more accurate. The performance and time consumption of a pairwise strategy should be between joint and pipeline models. Unfortunately, the performances of all pairwise models listed above are significantly below the best performing method in the BioNLP2009 challenge.

Van Landeghem et al. [54] emphasize the precision while tuning the parameters of classifiers with respect to the F_1-score. They determine the class of a trigger in two steps: first, they used dictionaries to create the candidate triggers along with the possible classes in parallel; second, they assign the event class with highest SVM score to the trigger at the end of event extraction. Single-argument events and multi-argument events are treated as different classes and the event extraction process is run two times to extract the recursive events. Móra et al. [55] use a handcrafted dictionary to generate the candidate triggers before pairwise detection. Using a dictionary to filter irrelevant words is an effective way to decrease the computational cost, but their handcrafted dictionary only cover 69.8% of the triggers, which may be a reason of their poor performance. They merged a C4.5 model and a handcrafted system to extract the pairs, and they defined very meticulous classes for the pairs, which contain true triggers but false arguments. Moreover, they only deal with (TRIGGER,PROTEIN) pairs. Ozgur and Radev [56] extracted pairs for

SVT, **BIND**, **REG** events separately by feature-based classifiers. However, they did not mention the details of the construction of multi-argument events and recursive events in their short paper. Apart from implementation issues, such as inefficient dictionaries and classifiers, the main problem of previous pairwise models is that none provides an effective way to solve the recursive events and multi-argument events.

2.5 Summary

In this chapter, we introduced the previous approaches used to solve the BioNLP Genia task. We described each step of the preprocessing pipeline because of the dependency between low-level and high-level NLP tasks, which are both required to solving the final problem. Previous works proved the importance of preprocessing methods and proposed successful solutions that achieve good performance. The best preprocessing solutions, which were used by most of participants, are highly task specified.

After preprocessing methods, we presented the main models for BioNLP that can be divided into three categories: joint models, pipeline models and pairwise models. Joint models consider all possible relations together and thus have a very high computing complexity but are very strong to capture global information. One of such models UCLEED achieves the best performance to the best of our knowledge. Pipeline models construct complex events using several steps, where each step successively build complex structures from simpler ones. Therefore, pipeline models have much lower complexity but will remove many potential candidates at the steps without global information. Pairwise models are trade-offs between joint and pipeline models; they create pairs instead of complex structure or fine-grained triggers. We suppose that this kind of model should perform better than pipeline models but worse than joint models. However, previous pairwise models are not well developed and actually obtain very bad performance. We present a well designed pairwise model in this thesis.

Apart from the problem modelization, previous works used different kinds of algorithms. Generally speaking, learning algorithms are better than the handcrafted rule-based systems. Comparing statistical learning methods and systems that learn rules from training sets, statistical learning methods outperform the rule-based systems except BioSEM that performed best on *Binding* event extraction. One reason of this situation is that symbolic rules are not powerful enough to capture the flexible and ambiguous natural language. Abacha and Zweigenbaum [57] show that combination of rule-based symbolic system and machine-learning system can improve the performance.

Chapter 3

A Pairwise Model

Pipeline models and joint models are two extremes. While pipeline models decompose the final extraction problem into basic atomic problems and solve them under strong independence assumptions, joint models try to solve the overall problem and consider all the possible dependencies between them. Pipeline models contain multiple steps, where each step makes predictions based on local information and predictions of previous stages, the main drawback is error cascading such as eliminating potential candidates too early in the process. A straightforward solution to overcome this issue is taking off the prediction and the extreme case follows this principle is joint model. Hence, we propose an intermediate model that blends the two steps of trigger detection and edge detection by relying on a pairwise structure. Indeed, we believe that pair structure captures essential interactions, enabling to detect triggers at the edge detection step. In this chapter, we propose a pairwise model that achieves very good performances, especially for extracting single argument events.

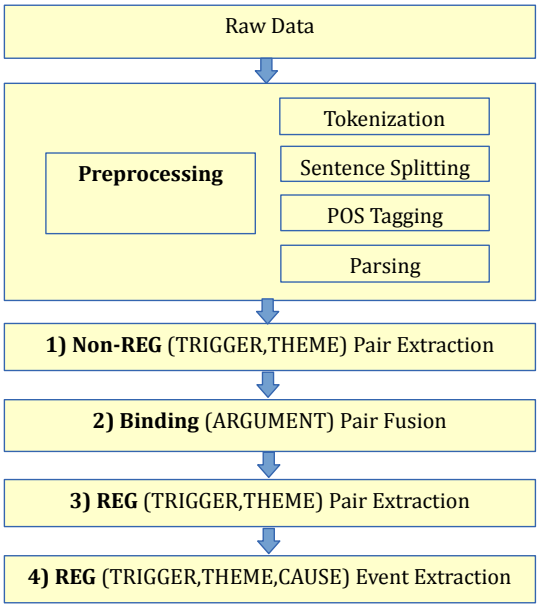


FIGURE 3.1: Pairwise extraction system

3.1 Introduction

In this chapter, we introduce the first model that we developed and used to participate the BioNLP 2013 Genia shared task. This is a pipeline system for which we tried to merge trigger classification and edge detection into a single step. The whole process is illustrated in Figure 3.1. This system splits (TRIGGER,THEME) detection into two steps for **Non-REG** events and **REG** events, due to the potential complexity of events, especially for recursive events. Multi-argument events such as *Binding* or **REG** are constructed after the necessary (TRIGGER,THEME) pair extraction step by either merging *Binding* THEME or assigning CAUSE ARGUMENT to **REG** events. This model facilitates inference compared to global models while relying on richer information compared to usual pipeline approach. The major drawback of this system is its inability to retrieve recursive **REG** events.

Apart from the model, we also studied the methods of combining output scores of binary classifiers for multi-class classification problems [58–60]. To address the multi-class problem, first, we used a series of binary SVMs in one-vs-rest framework, where hyper-parameters are tuned by cross-validation for each SVM individually. In addition to the SVMs, we added logistic regression transform to map SVM scores to the $(0, 1)$ interval. In the end, a set of thresholds was optimized to maximize the total micro-average F_1-score on the whole training set.

Following previous work, we created effective features focusing on three aspects: dependency parsing, tokenization and knowledge-base data. The crucial role of dependency features in BioNLP event extraction tasks is emphasized by [29, 42]. Therefore, we conducted a careful optimization of the dependency path features. We noticed that trimming was a very useful technique used in previous works including both symbolic and statistical approaches. Thus, we trim specific dependency paths with respect to their length, and get significant improvement over it. Besides, we studied the specific tokenization used by Stanford biomedical event parser, TEES and UCLEED systems,

which has proven to outperform the support tokenization when accompanied with their biomedical specified parser. We followed their work and additionally designed specific features based on support tokenization. In the end, protein names were abandoned in previous works because they are pointless to an event without background knowledge. Finally, following the EVEX system [38, 61] that integrated features from the knowledge base into the event classifier, we also conducted some work to extract potential events features from the IntAct knowledge base using protein names as query.

Because of an implementation error, our system only ranked sixth on official test, but after fixing the bug, it slightly outperforms the winner of BioNLP 2013 shared task. Besides, our system is less efficient for extracting *Binding* events, which we think is mainly due to the effectiveness of *Binding* THEME fusion step. However, it is independent to our major pair extraction step and can be reinforced additionally. The major drawback of this system is that it cannot extract recursive **REG** events.

3.2 Problem Modelization

Compared to the large body of work on pipeline models, pairwise models have been little studied. A few studies [54–56] report the basic pairwise idea. However, they do not give many details of how to construct events from pairs, and usually make a meticulous class system for different pairs. For example, Van Landeghem et al. [54] treat *Binding* events with different number of arguments as different classes and uses different dictionaries and classifiers to extract them. Different classes were created for the same event type regarding the number of argument and the type of argument (protein or other event). They solved the multi-class classification problem by using completely independent classifiers where each classifier uses different features. Moreover, they did not introduce how they deal with the conflicts arising when combining results of these various classifiers. All those pairwise approaches do not simplify the problem, but make it more complex compared to pipeline models or joint models. In this section, we describe our formulation of pair structure extraction and our decomposition of BioNLP Genia task into series of pairwise relation extraction problems.

3.2.1 Formulation of Pair Extraction

We denote $\mathcal{T}_S = \{t_i\}_i$ the set of candidate entities, $\mathcal{A}_S = \{a_j\}_j$ the set of candidate arguments in a given sentence S, and the set of event type labels is denoted $\mathcal{Y}$. The first step of a pipeline model assigns labels to candidate entities $t \in \mathcal{T}_S$. Instead, our pairwise model addresses the problem of classifying candidate trigger-argument pairs $(t_i, a_j) \in \mathcal{T}_S \times \mathcal{A}_S$. Denoting f_k the binary classifier predicting the label $k \in \mathcal{Y}$, pairwise extraction is performed by

$$\forall (t_i, a_j) \in \mathcal{T}_S \times \mathcal{A}_S, \quad \hat{y}_{ij} = \arg\max_{k \in \mathcal{Y}} f_k(t_i, a_j),$$

where $\hat{y}_{ij}$ predicts the event type of the pair made of the candidate trigger entity t_i and the argument a_j, an event being actually extracted when $\hat{y}_{ij} \neq \mathit{None}$.

This formulation predicts the event trigger and its argument jointly, so that the features of a candidate trigger entity t_i can help recovering the class of (t_i, a_j) and the features

of the path between t_i and a_j can help recovering the trigger status of t_i. It is similar to the UCLEED model, which judges the validity of an event through the sum of scores of trigger and edge detectors under constraints. However, in UCLEED, the score of edge detection only helps to determine the presence of event: when an event is detected, its type is only determined by the features of the candidate trigger. This seems to be a reasonable process, since arguments should not change event types. However, due to the complexity of human language, uncertainty of preprocessing and imperfect representation methods, a conceptually reasonable process may not necessarily lead to a better practical performance. Our method gives the classifier the opportunity to use trigger features along with edge features to determine the type of trigger, which is simpler but also quite robust.

3.2.2 Problem Decomposition

As for typical pipeline approaches, we present here a system where each step generates examples based on the output of previous steps. The major difference between our approach and pipeline systems is that we fuse the detections of triggers and edges. We recall that in pipeline models, edge detection aims to classify candidate edges as THEME edges, CAUSE edges or *None*. As described in Section 1.1.3, only **REG** events could have CAUSE argument and such events are possibly recursive since their arguments can be any other events. If we had merged the detection of edge types and trigger types into a single classification problem, all the connections between candidate entities could have been considered and many nonviable choices could have been created. As a result, we only tackle the (TRIGGER, THEME) pair extraction and refine this problem into **Non-REG** event extraction and **REG** event extraction. In our framework, we do not perform (TRIGGER, CAUSE) pair extraction, but rather treat CAUSE as an additional argument to an existing (TRIGGER, THEME) pair. In our pairwise model, the structured information extraction problem is decomposed into four sub-problems, which can be solved by a sequence of traditional classifiers.

(1)	**Non-REG** (TRIGGER, THEME) pair extraction	**Non-REG** Event Extraction
(2)	**BIND** THEME fusion	
(3)	**REG** (TRIGGER, THEME) pair extraction	**REG** Event Extraction
(4)	**REG** CAUSE assignment	

It is thus similar to a pipeline system, except that the first and third steps perform pairwise extractions.

3.2.2.1 Non-REG Event Extraction

As defined in Section 1.1.3, **Non-REG** events include single argument events (**SVT**) and *Binding* events, which can involve a second THEME argument. Extracting (TRIGGER, PROTEIN) pairs directly solves the **SVT** event extraction problem and constructs preliminary pairs for *Binding* events. As many pipeline systems do, we added a post-processing step to combine the multiple arguments of Binding events after pairwise extraction.

Non-REG (TRIGGER, THEME) Pair Extraction In the first step of our model, candidate trigger entities are constructed using a dictionary-based string match. The dictionary is generated from the gold annotations, in which we store sequences of tokens

encoding entities together with their sub-tokens. We also stemmed the last token in entities and manually eliminated stop words such as "to", "are", "of". Using this dictionary, we recursively build the candidate trigger entity set $\mathcal{T}_S$ by adding the longest token sequence that matches the dictionary. Following the formulation of Section 3.2.1, we obtain the (t_i, a_j) pair set with $\mathcal{A}_S$ containing all the proteins in sentence and label set $\mathcal{Y} = \{$*Gene_ expression*, *Transcription*, *Localization*, *Phosphorylation*, *Protein_ catabolism*, *Binding*, *None* $\}$.

BIND THEME Fusion When more than two pairs (t_i, a_j) with the same candidate trigger entity t_i are extracted and labeled as *Binding* by the previous step, we create all combinations $\{(t_i, a_j, a_k)|j \neq k\}$ as potential *Binding* events and evaluate them with a dedicated classifier. Once a combination (t_i, a_j, a_k) is predicted as being true, $\hat{e}_{ij} = (t_i, a_j, Binding)$ and $\hat{e}_{ik} = (t_i, a_k, Binding)$ are replaced by $\hat{e}_{ijk} = (t_i, a_j, a_k, Binding)$. Note that in this step, we actually take the features between two proteins a_j, a_k into account because other pairwise relations are observed in last step. Compared to joint models, a major shortcoming of our approach is that it does not consider pairs jointly from the start. As a result, the detection of a partial *Binding* event does not encourage the detection of additional arguments. For example, consider a *Binding* event compose of two pairs, where (t_i, a_j) is easy to detect whereas (t_i, a_k) is hard. Our approach will fail to extract $\{(t_i, a_k)$ even if (a_j, a_k) are very likely belong to the same event. This could be fixed by using only the detected *Binding* trigger instead of detected pairs from the previous step, that is by creating the set of triplets $\{(t_i, a_j, a_k)|j \neq k, a_j, a_k \in \mathcal{A}_S\}$ where t_i comes from any extracted *Binding* pairs, and $\mathcal{A}_S$ contains all the proteins in sentence. However, since distances between two arguments are generally long in the dependency parse, we doubt about the effectiveness of this scheme. Although this step deals with triplet (t_i, a_j, a_k), we only consider the relation between (a_j, a_k) since other features are already observed in the previous step. Thus, the detection of multi-argument *Binding* events is actually the detection of valid protein pairs. During training, the example set contains all the possible protein pairs when a *Binding* trigger presents in the sentence. Note that, proteins, which do not connect to any *Binding* triggers, are observed in training set. It is a very coarse solution that might be the reason of our relatively lower **BIND** performance.

3.2.2.2 REG Event Extraction

When all **Non-REG** events have been extracted, there are two possible strategies to build **REG** events. The first one consists in testing every possible triplet combination $(t_\alpha, \text{THEME} : a_\beta, \text{CAUSE} : a_\gamma)$ by adding all the proteins and extracted triggers into the candidate argument set $\mathcal{A}_S$. Compared to a pairwise scheme, triplet structure leads to considerably larger candidate example sets. Another strategy is to decompose the triplet prediction into two steps: (1) predict $(t_i, \text{THEME} : a_j)$ pairs and (2) assign the CAUSE argument. This method lowers the number of candidate examples, but it postpones the prediction of CAUSE arguments after the prediction of **REG** triggers, which makes using **REG** triggers as CAUSE argument possible. We selected the second strategy driven by the preference of capturing recursive events and of its lower computing complexity.

REG-THEME Pair Extraction In order to extract **REG**-THEME pairs based on previous extractions, we followed the pairwise approach, while redefining the candidate

trigger entity set and candidate argument set. Denote $\mathcal{T}_S^{\textbf{Non-REG}}$ the candidate trigger entity set in **Non-REG** event extraction step, $\mathcal{E}_S^{\textbf{Non-REG}} = \{\hat{e}_i\}$ the extracted events in previous steps where candidate entity t_i had been indicated as trigger in an event $\hat{e}_i$. The new set of candidate trigger entities collects the remaining candidate trigger entities $\mathcal{T}_S^{\textbf{REG}} = \mathcal{T}_S^{\textbf{Non-REG}} - \{t_i | \exists \hat{e}_i \in \mathcal{E}_S^{\textbf{Non-REG}}\}$ and the set of arguments collects all the proteins, together with the extracted events $\mathcal{A}_S^{\textbf{REG}} = \mathcal{A}_S^{\textbf{Non-REG}} \bigcup \{t_i | \exists \hat{e}_i \in \mathcal{E}_S^{\textbf{Non-REG}}\}$. The label set naturally becomes $\mathcal{Y} = \{$*Regulation*, *Positive_ regulation*, *Negative_ regulation*, *None* $\}$. The pairwise formulation remains

$$\forall (t_i, a_j) \in \mathcal{T}_S \times \mathcal{A}_S, \hat{y}_{ij} = \arg\max_{k \in \mathcal{Y}} f_k(t_i, a_j).$$

Note that our approach cannot retrieve the recursive events that involve **REG** events in their THEME argument.

REG-CAUSE Assignment The **REG**-CAUSE assignment step seeks arguments for extracted **REG** events $\mathcal{E}_S^{\textbf{REG}}$. Considering event e_{ij} as an entity, assigning CAUSE argument can be seen as a nested pairwise extraction step. By establishing candidate argument set $\mathcal{A}_S^{\text{CAUSE}} = \mathcal{A}_S^{\textbf{REG}} \bigcup \{t_i | \exists e_i \in \mathcal{E}_S^{\textbf{REG}}\}$, we can build the candidate pair set $\{(e_{ij}, a_k)\} = \mathcal{E}_S^{\textbf{REG}} \times \mathcal{A}_S^{\text{CAUSE}}$. We call it a nested pairwise step because the candidate pair (e_{ij}, a_k) is actually a triplet $(t_i, \text{THEME} : a_j, \text{CAUSE} : a_k)$ in this case.

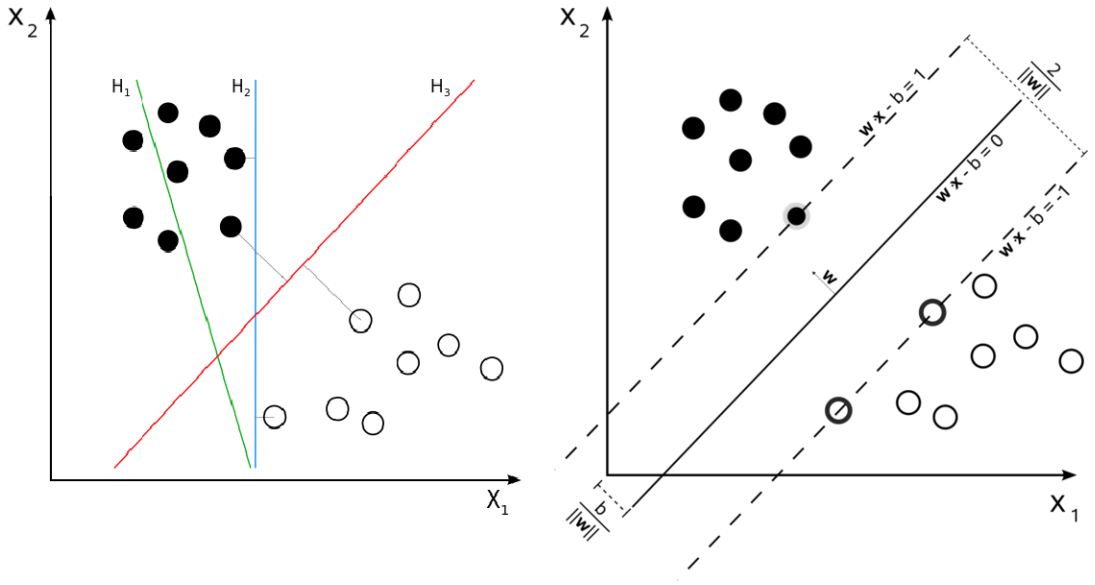
3.3 Implementation

In this section, we introduce the implementation details of our model. As described in the previous section, the complex event extraction problem is decomposed into four steps, where each step involves a binary or multi-class classification problem. Section 3.3.1 describes the classifier designed for multi-class classification, Section 3.3.2 lists our features together with our exact preprocessing and corresponding hyper-parameters.

3.3.1 Classifier

Two types of classification tasks appear in the four steps of our system displayed in Figure 3.1. The two pairwise extraction steps (steps 1 and 3) amount to solving multi-class classification problems. The two other steps (steps 2 and 4) amount to binary classification problems. Multi-class classification is the problem of classifying instances into more than two classes.

In our work, we paid particular attention to multi-class classifier solutions, and we developed a classification protocol dedicated to our problem. It is based on binary SVM classifiers followed by a combination of scores aiming at optimizing the overall micro-average F_1-score. We ran many experiments testing many variants to choose the best classification protocol, but we only describe here the solution that was eventually selected in our system. Some of the other devised variants are described in Chapter 5.



(A) Candidate hyper-planes (B) SVM separating hyper-plane

FIGURE 3.2: SVM separating hyper-plane

3.3.1.1 SVM for Cost-Sensitive Multi-Class Classification

In this section, we introduce our main classifier, which are cost-sensitive SVMs in the one-vs-rest framework, which form the first layer of our overall classifier.

Standard SVM Binary Support Vector Machines (SVM) are supervised learning models that are often used for classification. Given a training set of examples labeled into two categories, a SVM training algorithm builds a model that assigns new example into one or the other category. A major characteristic of SVM is that the model tries to separate the classes by a margin that is as wide as possible. Though SVM can be extended to nonlinear discrimination thanks to kernels, we only used linear SVMs due to the huge size of the feature space.

Given a data-set $\mathcal{D} = \{\mathbf{x}_n, y_n\}$, where $\mathbf{x}_n$ is the feature vector and $y_n \in \{-1, 1\}$ is the label for example n, linear SVM aims to find hyper-plane that perfectly separates the classes. If the data-set is linearly separable, several hyper-planes allow for this separation, as shown in Figure 3.2a. The hyper-plane sought by SVM is the one that maximizes the minimum distance to the examples of the data-set. The SVM score for point n is $\mathbf{w} \cdot \mathbf{x}_n - b$. Using the convention that points at the margin have a score in $\{-1, 1\}$, their distance to the separating hyper-plane is $\frac{2}{\|\mathbf{w}\|}$ (see Figure 3.2b). For computational convenience, this distance is maximized by minimizing $\frac{1}{2}\|\mathbf{w}\|^2$. Stating that all the points are out of the margin reads:

$$\forall n \in \{1, \ldots, N\}\,, \quad y_n(\mathbf{w} \cdot \mathbf{x}_n - b) \geq 1 \quad .$$

So the optimization problem becomes:

$$\begin{aligned} &\arg\min_{\mathbf{w},b} && \tfrac{1}{2}\|\mathbf{w}\|^2 \\ &\forall n \in \{1,\dots,N\}\,, && y_n(\mathbf{w}\cdot\mathbf{x}_n - b) \geq 1 \ . \end{aligned}$$

To accommodate for non-separable classes, soft margins modify the objective function by introducing non-negative slack variables $\{\xi_n\}_{n=1}^N$, that measures how the examples $\{\mathbf{x}_n\}_{n=1}^N$ violate the margin constraint. The optimization problem becomes:

$$\arg\min_{\mathbf{w},b,\{\xi_n\}_{n=1}^N} \quad \tfrac{1}{2}\|\mathbf{w}\|^2 + C\sum_{n=1}^N \xi_n \tag{3.1}$$

$$\forall n \in \{1,\dots,N\}\,, \quad y_n(\mathbf{w}\cdot\mathbf{x}_n - b) \geq 1 - \xi_n \ . \tag{3.2}$$

The objective function allows to compromise between the margin size associated to $\|\mathbf{w}\|$ and the data misfit $\sum_{n=1}^N \xi_n$, tolerating mislabeled examples. This trade-off is controlled by the hyper-parameter C, where smaller C values lead to less accurate fitting of training examples.

Once the model has been trained, the SVM predicts the class $\hat{y}$ of example $\mathbf{x}$ by thresholding the SVM score:

$$\hat{y} = \text{sign}(\mathbf{w}\cdot\mathbf{x} + b) \ .$$

SVM with Asymmetric Costs In our application classes are highly unbalanced, as shown in Table B.5 and B.4, which list the number of events of different classes. Furthermore, we are targeting the maximization of F_1-scores, which usually differs from the usual minimization of the misclassification error. We thus used different losses for positive and negative examples. Following [62, 63], this results in two hyper-parameters (C^+/C^-) that are tuned by cross-validation. As mentioned in Equation 3.1, C controls the trade off between margin and training errors. The two hyper-parameters C^+/C^- act as C for positive and negative examples respectively. The objective function becomes:

$$\frac{1}{2}\|\mathbf{w}\|^2 + C^+ \sum_{\{n|y_n=+1\}} \xi_n + C^- \sum_{\{n|y_n=-1\}} \xi_n \ .$$

One-vs-Rest Framework A binary SVM does not directly address our multi-class problem, but it can be a compound of a multi-class classifier in the one-vs-rest framework. In this framework, assuming $K > 2$ exclusive classes, K binary classifiers are trained. They all use the same training data, but with different objectives. Classifier $k \in \{1,...,K\}$ is trained using positive labels for the examples of class k, and negative labels for all the other examples. The resulting scores $\{f_k\}_{k=1}^K$ thus indicate the likeliness of each class alone. Finally, a single class is predicted by selecting the class with maximal score:

$$\hat{y} = \arg\max_{k\in\{1,\dots,K\}} f_k(\mathbf{x}) \ . \tag{3.3}$$

3.3.1.2 Training Procedure

Notation We denote by $\mathbf{X}$ the $N \times M$ matrix of training examples, where N is the number of examples, and M is to the size of feature vector. The training labels form a

$N \times K$ matrix $\mathbf{Y}$, where K is the number of classes, and $\mathbf{Y}(n,k) = 1$ if $\mathbf{x}_n$ belongs to class k, and $\mathbf{Y}(n,k) = -1$ otherwise. The binary class label for class k is the kth column of $\mathbf{Y}$, denoted by $\mathbf{Y}^k$.

Setting C^+/C^- Hyper-Parameters For each event detection problem, we selected the C^+/C^- values of the corresponding binary SVM, so as to maximize the F_1-score of the corresponding event type taken in isolation. Since we calibrate the SVM scores in the following combination step, this selection is based on the thresholded SVM scores as follows.

The SVM scores $f_k(\mathbf{x}) = \mathbf{w}_k \cdot \mathbf{x} + b_k$ are used to produce the class assignments

$$\hat{y}^k(f,\theta) = \text{sign}(f_k(\mathbf{x}) + \theta) \ ,$$

where θ is a candidate offset. On the overall training set, the offset is optimized by maximizing the F_1-score computed from the true labels $\mathbf{Y}^k$ and the predictions $\hat{\mathbf{Y}}^k(\mathbf{F}^k, \theta))$, where $\mathbf{F}^k$ is the N-dimensional vector of SVM scores. This optimization is applied in the cross-validation process detailed in Algorithm 1, on the holdout SVM scores, so as to select the C^+/C^- values.

Algorithm 1 Estimation of the Optimal C^+/C^- for the Binary SVMs

input training examples $\mathbf{X}$, training labels $\mathbf{Y}$, candidate hyper-parameters $\Lambda = \{\lambda\}$, number of cross-validation folds Q

1: **for** $k = 1 \rightarrow K$ **do**
2: **for each** $\lambda \in \Lambda$ **do**
3: compute the holdout SVM scores $\mathbf{F}^k(\lambda)$ by Q-fold cross-validation for class k
4: **if** $max_\theta F_1(\mathbf{Y}^k, \hat{\mathbf{Y}}^k(\mathbf{F}^k, \theta))$ is optimal **then**
5: **store** $\hat{\lambda}^k \leftarrow \lambda$
6: **end if**
7: **end for**
8: **end for**
9: **return** optimal hyper-parameters $\{\hat{\lambda}^k\}_{k=1}^K$

SVM Scores Combination Empirically, it has been observed that the one-vs-rest framework provides good solutions to the multi-class problem, but theory shows that this scheme is not consistent, so that it should be used with care. In particular, there is no reason to believe that the scores f_k in (3.3) are commensurate, and thus that the majority vote makes sense. This problem can be somewhat circumvented by transforming the SVM scores into pseudo-probabilities [58–60], but this strategy may not be suited for optimizing the F_1-score. Here, we deviate from this strategy by tuning decision thresholds on the normalized SVM scores: we do not look at reliable pseudo-probabilities, but we use the logistic function to normalize the SVM scores and to ease the search of the decision thresholds.

Logistic Regression Unlike SVM, logistic regression is a probabilistic model that estimates posterior probabilities of classes. It usually refers to binary logistic regression,

whereas multinomial logistic regression directly solves multi-class problems. A key element of logistic regression is the logistic function, which takes its values in $(0, 1)$. Here, we used the logistic transform to map each SVM score to the $(0, 1)$ interval, so as to set a well-grounded common scaling for these scores.

Given an example $\mathbf{x}$ described by its K-dimensional vector of SVM scores $\mathbf{f}$, we transform these scores as

$$g_k(\mathbf{x}) = \frac{1}{1 + \exp -(\boldsymbol{\beta}_k \cdot \mathbf{f} + \alpha_k)} \ ,$$

whose parameters are adjusted by minimizing

$$C\|\boldsymbol{\beta}_k\|^2 + \sum_{n=1}^{N} \log(1 + \exp(-y_n \cdot g_k(\mathbf{x}_n))) \ ,$$

where C is a regularization hyper-parameter, which was set to a small common value to avoid numerical instability.

Decision Thresholds The BioNLP shared tasks assess the performances of event detections by the micro-average F_1-score for each sub-group of categories and for the overall task. We thus optimized the decision thresholds so as to maximize these micro-average F_1-scores. We recall that for each class k, one can calculate the precision $prec_k = \frac{tp_k}{tp_k + fp_k}$ and recall $rec_k = \frac{tp_k}{tp_k + fn_k}$, where $tp_k = |\{n | y_n^k = 1, \hat{y}_n^k = 1\}|$ is the number of true positive predictions, $fp_k = |\{n | y_n^k = 0, \hat{y}_n^k = 1\}|$ is the number of false positive predictions and $fn_k = |\{n | y_n^k = 1, \hat{y}_n^k = 0\}|$ is the number of false negative predictions. For all the classes, the micro-average precision and recall are

$$prec = \frac{\sum_{k=1}^{K} tp_k}{\sum_{k=1}^{K} tp_k + \sum_{k=1}^{K} fp_k}$$

$$rec = \frac{\sum_{k=1}^{K} tp_k}{\sum_{k=1}^{K} tp_k + \sum_{k=1}^{K} fn_k} \ .$$

The micro-average F_1-score is the harmonic mean of micro-average of precision and recall:

$$F_1 = 2 \cdot \frac{prec \cdot rec}{prec + rec} = \frac{2 \cdot \sum_{k=1}^{K} tp_k}{2 \cdot \sum_{k=1}^{K} tp_k + \sum_{k=1}^{K} fn_k + \sum_{k=1}^{K} fp_k} \ .$$

Compared to the macro-average F_1-score, which is the mean of F_1-scores of each class, the micro-average F_1-score underrates the impact of small classes.

The last step of our training process consists in setting decision thresholds $\Theta = (\theta_1, \ldots, \theta_K)$ on the output g_k of the logistic transform, to compute the final decision $\hat{y} = \arg\max_k(g_k + \theta_k)$. The maximization with respect to Θ of the micro-average F_1-score on the entire training set is hard, since the objective function is discontinuous. There are however

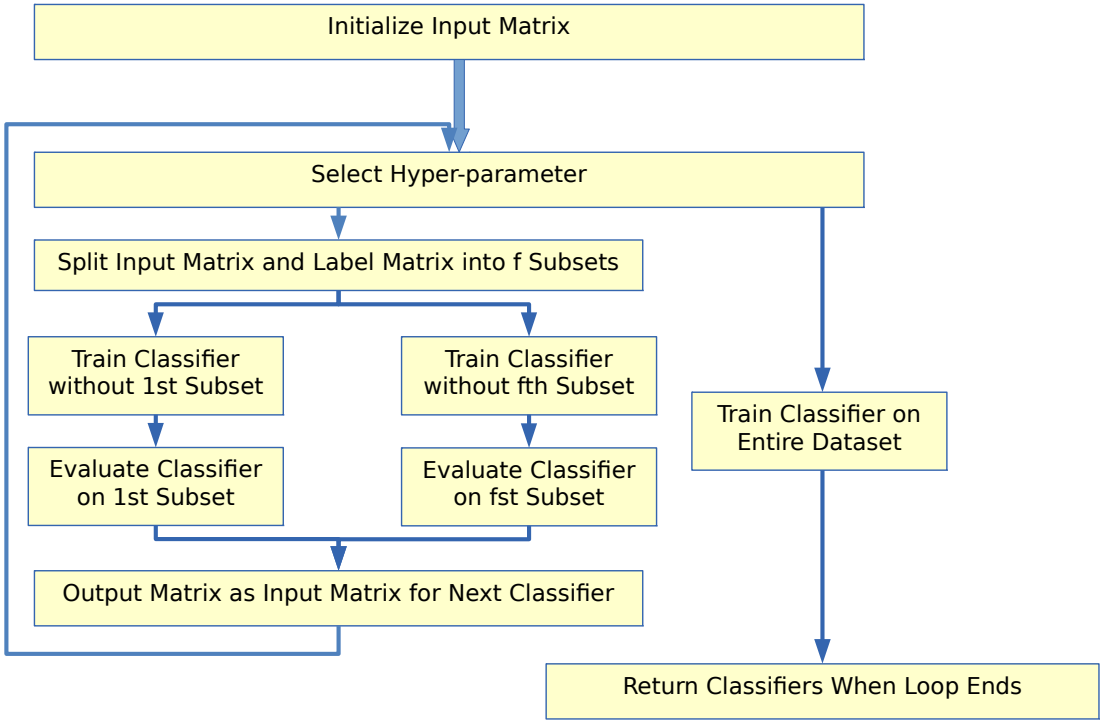


FIGURE 3.3: Classifier Chain

numerical methods that converge towards a local optimum. Here, we optimize iteratively each θ_k by Brent's method [64], which is a root-finding algorithm. As shown in Algorithm 2, each iteration optimizes $F_1(\mathbf{Y}, \hat{\mathbf{Y}}(\mathbf{G}, \Theta))$ with respect to a single threshold θ_k, while other ones are fixed. We did not set a stopping criterion and simply limited the number of optimization steps.

Algorithm 2 Optimization of the Classification Thresholds

input Classifier scores $\mathbf{G}$, training labels $\mathbf{Y}$, number of iteration R.

1: **initialize** thresholds $\Theta = (\theta_1, \ldots, \theta_K)$
2: **for** $r = 1 \rightarrow R$ **do**
3: **for** $k = 1 \rightarrow K$ **do**
4: **update** $\theta_k = \arg\max_{\theta_k} F_1(\mathbf{Y}, \hat{\mathbf{Y}}(\mathbf{G}, \Theta))$
5: **end for**
6: **end for**
7: **return** optimized thresholds Θ

Classifier Chain Figure 3.3 shown how we train the classifier chain. The whole process is a classifier chain composed by a list of classifiers and their candidate hyper-parameters $\{(cls_i, \Lambda_i)\}$, where cls_i denote the ith classifier and Λ_i the candidate hyper-parameters. Output scores of preceding classifier are provided to succeeding classifier. In order to simulate the test environment, output matrix is created through cross-validation during training. Algorithm 3 illustrates the exact training procedure. The first input matrix is initialized by the matrix of training examples $\mathcal{X}$. For each classifier cls_i, we first select the optimal hyper-parameter given the entire input matrix. Then we split the

Algorithm 3 Classifier chain

input training examples $\mathbf{X}$, training labels $\mathbf{Y}$, number of cross-validation fold Q, a list of classifiers along with their candidate hyper-parameters $cL = \{(cls_i, \Lambda_i)\}$

initialize input mat $\mathbf{T} = \mathbf{X}$
for each $(cls_i, \Lambda_i) \in cL$ **do**
  initialize output matrix $\mathbf{O} = \emptyset$
  select the optimal hyper-parameter $\hat{\lambda}_i$ from Λ_i for cls_i
  split input matrix and labels into Q sub-sets $(\mathbf{T}, \mathbf{Y}) = \bigcup_q (\mathbf{T}_q, \mathbf{Y}_q), q = 1 \dots Q$
  for $q = 1 \rightarrow Q$ **do**
    train cls_i on $(\mathbf{T}, \mathbf{Y}) - (\mathbf{T}_q, \mathbf{Y}_q)$
    evaluate cls_i on $(\mathbf{T}_q, \mathbf{Y}_q)$ to get output score $\mathbf{S}_q$
    update output matrix $\mathbf{O} = \mathbf{O} \bigcup \mathbf{S}_q$
  end for
  fit classifier cls_i on $(\mathbf{T}, \mathbf{Y})$
  update input matrix $\mathbf{T} = \mathbf{O}$
end for
return trained classifiers $\{cls_i\}$

input matrix and corresponding label into f subsets. In order to avoid over-fitting, the output matrix $\mathbf{O}$ is generated through a cross-validation process, where output score of each subset is generated by the classifier trained on other subsets. Classifier cls_i is then trained on entire input matrix. As described before, the output matrix is assigned as input matrix for next classifier at the end of each loop. Finally, a list of trained classifiers is returned by the algorithm.

3.3.2 Feature Study

Pipeline systems from [29, 42] only differ in their features. We believe the improvement in F_1-score from 30 to more than 50 is mostly due to good feature engineering. In fact, most of current natural language processing approaches are simple combinations of linear models based on powerful features.

Our implementation somewhat follows many heuristics developed in the UCLEED system. As in UCLEED, we use the head token, which is selected heuristically, to represent an entity. Therefore, an entity is described by the features of its head token such as stemmed word, POS, prefix, *etc.* The dependency path between head tokens is used as an alternative to the dependency path between entities. Besides, we used different strategies for low-level preprocessing and introduced a simple method to improve the quality of the dependency path. Finally, we took semantic information from a knowledge base into account as well.

3.3.2.1 Multiple Tokenizations & Sentence Splitting

Most NLP practical approaches follow from the theories studied in linguistics and solve the high-level problems in a bottom-up way. We pursue a high-level information extraction task by relying on many low-level NLP tasks: word segmentation, sentence breaking, part-of-speech (POS) tagging, parsing. Some event extraction methods also

involve named entity recognition. Those tasks form a big pipeline and different low-level solutions typically impact the results of the high-level tasks. Since we are not primarily interested in feature engineering, we mostly re-used features and heuristics from previous works. Support tokenization and Stanford tokenization were selected as tokenizers: the first is provided by BioNLP task organizers, the second was used by the winner of BioNLP 2011 shared task.

Support Tokenization It is the open access tokenization from the BioNLP task, which was generated by the GTB-tokenize.pl script that attempts to mimic the tokenization used by the Genia Treebank. As mentioned in Section 2.2.1, the major characteristic of support tokenization is that it conserves compound words, slashes, *etc.* From a linguistic viewpoint, this is a good tokenization for human to read because one can easily decodes the inner structure of compounds word and splits parallel items concatenated by slashes: it makes the sentence easier to understand. We also believe that sentences using this tokenization should be easier to parse. For example, extracting relations in "A and B/C/D" is much simpler than extracting relations in "A and B / C / D", where in the first case "B/C/D" is treated as one token. However, this nice tokenization is not perfectly suitable for BioNLP Genia task because some events need the decomposition of the compound words to extract relations between the finer-grained objects.

Stanford Tokenization It appears in McClosky's BioNLP event parser [17], and was used by UCLEED, TEES and TEES 2.1 [18–25]. Unlike support tokenization, this algorithm breaks specific compound words and slashes with handcrafted heuristics. As discussed in Section 2.2.1, this algorithm improved performance in previous works. A possible reason is that breaking compound words improves the sparsity of feature space and enables higher level parsing to provide useful task specific features. Note that the head token of a candidate trigger entity or argument is selected based on this fine-grained tokenization. Most of our features are based on this fine-grained tokenization, but we did not investigate the reasons of the advantages of this tokenization.

Longest Sentence We needed to define a heuristic when the two tokenizations were not consistent. Documents have to be cut into sentences before further parsing. The two parsers break sentences in different ways, so that we had to resolve conflicts in order to use them in the same framework. Assuming that longer sentences increase the chances of detecting events, we simply connected separated sentences when breaking points were incompatible.

Coarse Tokenization Features An entity is described by the features of its head token, which is defined by the fine-grained Stanford tokenization. But we also used some features from the coarse support tokenization, which can bring additional information, in particular to detect compound words. Given a head token of candidate trigger entity based on fine grained tokenization, we think it is useful for the classifier to know whether this head token is near a protein or belongs to a compound word. Therefore, two features were developed based on support tokenization: one indicates whether the head token of candidate trigger entity belongs to a word that contains a protein; another indicates whether the head token of a candidate trigger entity and its candidate argument are in the same word.

3.3.2.2 Dependency Path Trimming

The dependency path features are the crucial features in relation extraction because they encode how tokens interact with each other. Since dependency parse represents a sentence by a tree or directed acyclic graph, people usually use the shortest path between two tokens in the dependency parse tree/graph to represent their relations. However, such dependency relations are usually presented in a tree or graph format, which cannot be understood effectively by current algorithms. For syntactic methods, Buyko et al. [32] report that trimming dependency trees provides more direct representative dependency relations between tokens. Reports of statistical methods always pass over this issue with the purpose of emphasizing successes of their statistical models. But when we studied the UCLEED system, we discovered that they heuristically removed some uninformative edges from the dependency path. We implemented similar strategies.

Encoding Paths As introduced in Section 2.3.2.1, current encoding methods transform the dependency path into bag of triplets, used as feature vector. We recall that *E-walk* and *V-walk* are two widely used methods for representing the dependency path by a bag of triplets. *E-walk* represents the dependency paths into bags of (*dep-tag*, *token*, *dep-tag*), whereas *V-walk* represents the dependency paths into bags of (*token*, *dep-tag*, *token*).

Given a dictionary that indexes all the triplets in the training set, a bag of triplets represents a set of triplets by a vector of counts, where each component of the vector indicates the number of occurrences of the indexed triplet in the set. Obviously, sequential information cannot be reconstructed from those encoding methods. Even if some short sequences can be reconstructed with the help of grammatical and/or semantic knowledge, there is usually no way to recover the original sentence for longer dependency paths, even for humans. In this case, different dependency paths could share the same bag of triplets representation, and those dependency paths could belong to examples in different classes. Therefore, examples with longer dependency paths have higher possiblities of sharing features with each other, regardless of their category. The features then become less discriminant. Moreover, bags of triplets produce large amount of features, for which slightly reworded phrases are hard to recognize. We thus assumed that only short dependency paths make reliable features for classification and that long paths should be discarded.

Statistics of Path Lengths Figure 3.4 represents the distribution of path lengths between TRIGGER and ARGUMENT for each event type and for the *None* classes in training phase of two (TRIGGER, THEME) pair extraction steps. Distances between most of (TRIGGER, THEME) pairs are smaller than five, whereas for *None*, the distribution is very flat. Examples do not have dependency path features when the path length is 0. To ignore the longer paths, the dependency path features of the examples that contain long paths are set to zero, resulting in the same features as the examples with null path lengths. Such group contains half of *None* examples and less than 10% **Non-REG** event examples, which reveals a strong feature to distinguish *None* examples from **Non-REG** examples: examples with null dependency path features are probably not events. *Binding* event has the worst proportion out of **Non-REG** events. We suppose that it causes *Binding* event more difficult to extract than other **Non-REG** events, despite multi-argument problem. **REG** events show better distribution than **Non-REG** events,

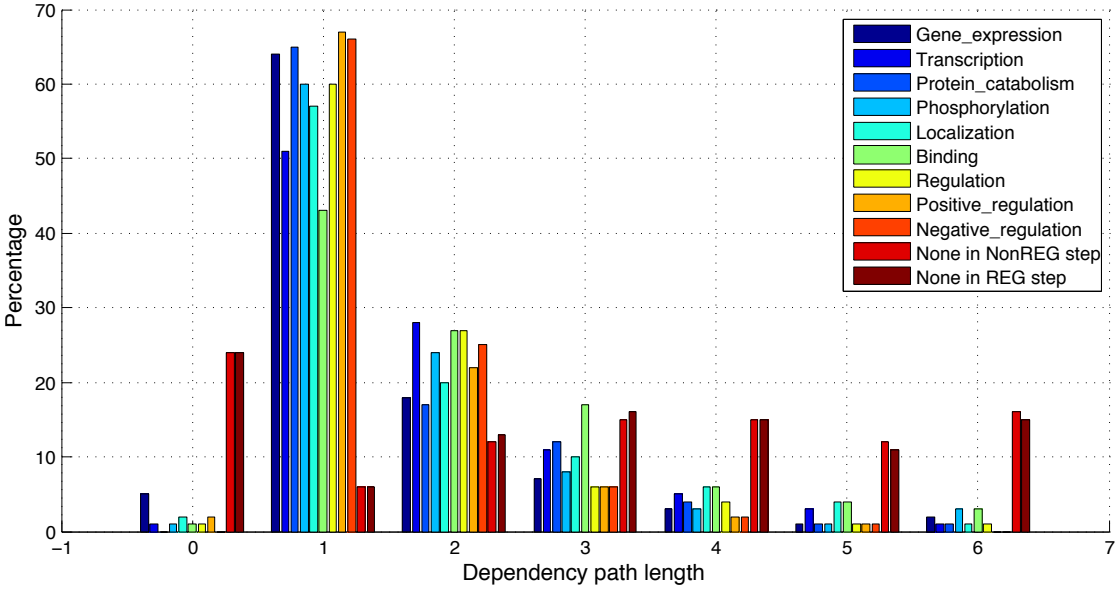


FIGURE 3.4: Frequency of examples according to the length of the dependency path between THEME and ARGUMENT

which is unexpected. Figure 3.5 shows the length distribution of dependency path between arguments for multi-argument events and the *None* examples during training phase of *Binding* THEME fusion and **REG**-CAUSE assignment. We see clearly that distances between two ARGUMENTs are much longer, which explains the difficulty of multi-argument events, especially for *Binding* event.

Similarly to the longest dependency paths, the shortest paths were shown to be harmful. It is easy to see that the features provided by the shortest paths are probably a part of features provided by the longer paths. If most of examples with the shortest paths are negative examples, it is difficult for a linear classifier to distinguish them. Accounting for these atypical paths may introduce too much variability. Thus, we discarded the paths with only one edge for **REG** events.

3.3.2.3 Knowledge Base

When experts read biomedical research articles, they sometimes have to build their interpretation upon their extensive background knowledge. Likewise, collecting the known interactions between proteins should provide a better chance to treat the ambiguities of natural language. We used the information available in publicly available knowledge bases to this end.

First, given the protein annotation, we retrieved all the protein names mentioned in the texts. Note that we did not process incomplete protein names such as "*3*" in "*IL-1/3*" for simplicity. Then, we obtained the map between protein names and protein IDs from the UniProt knowledge base [1]. Since the protein name strings were usually an imperfect match, we collected the top 5 IDs for each protein name string. Protein name strings with exactly the same top 5 IDs were treated as different representations of the same

[1] Available from `uniprot.org`.

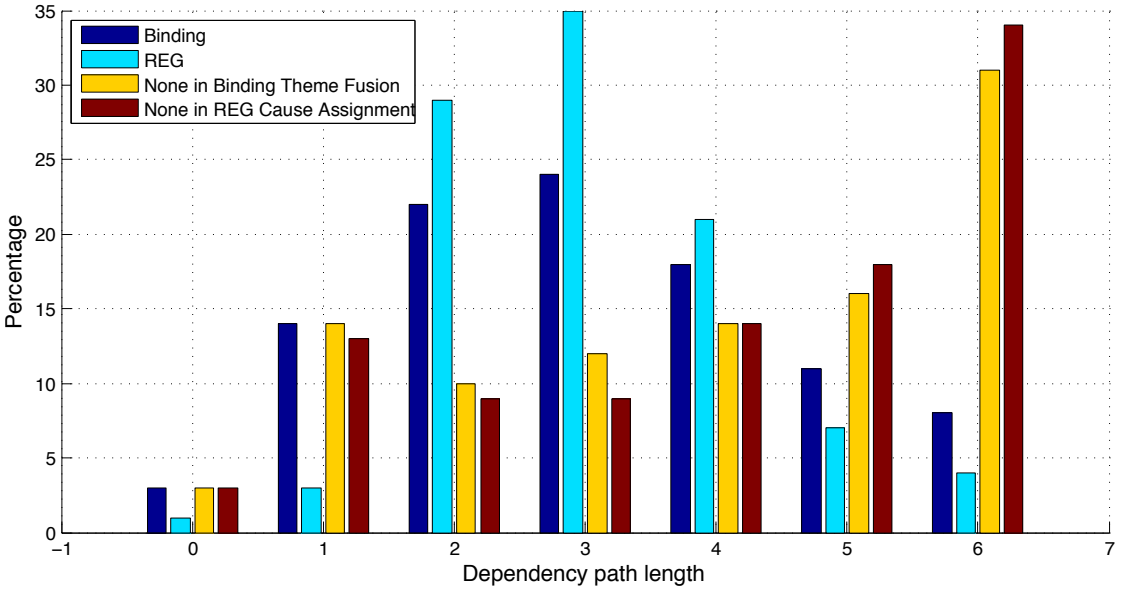


FIGURE 3.5: Frequency of examples according to the length of the dependency path between two ARGUMENT tokens in multiple argument events

protein, so that interaction knowledge of those IDs was shared by these protein names. Finally, we took the protein interaction information from the IntAct knowledge base [2]. For each recorded protein, we collected the events it participated to, and the recorded proteins it interacted with. This knowledge was used in both (TRIGGER, ARGUMENT) and (ARGUMENT, ARGUMENT) extraction. In (TRIGGER, ARGUMENT) pair extraction, the possible event types of the protein arguments retrieved from the knowledge were used as features. In multi-argument events, the information was used to indicate the known interactions between two protein arguments.

3.3.2.4 Feature Summary

We list all the features we used in Table 2.1 in Section 2.3. All (TRIGGER, THEME) pair extraction steps use all features except the one that is only used between two arguments. The *Binding* THEME fusion step does not use features for candidate trigger entity and **REG** CAUSE assignment step only uses the stemmed token as argument feature.

3.4 Experiments

To participate to the BioNLP 2013 shared task, we made lots of experiments on BioNLP 2011 and BioNLP 2013 training and development data-sets in order to find the optimal feature settings. In this section, we introduce the detailed results of different feature configurations as well as our performances on the test set of both BioNLP 2011 and 2013 shared tasks.

[2] Available from www.ebi.ac.uk/intact.

3.4.1 Feature Adjustment

As mentioned in Section 3.3.2, we developed new features and some of them contain hyper-parameters to adjust. Since each step of our system solves a different problem, it is necessary to find the optimal feature setting at each step. However, searching the best feature settings is complex because performances of succeeding steps rely on preceding ones and this impact is hard to evaluate. We mostly focused our feature adjustment on the trimming of the dependency path and have run experiments to define the best possible configuration. For the other features, we tried limited number of settings or followed previous works. All results reported in this section have been obtained using evaluations on the development set of BioNLP 2013 shared task.

3.4.1.1 Dependency Path Trimming

Since each step of our system requires features based on the dependency path, this makes selecting thresholds for trimming dependency paths a difficult problem. A reliable way is to select the best thresholds with respect to the performance on the development set. However, the dependency between **REG** and **Non-REG** events induces that the final score is not a sum of scores at each step. Hence, finding a perfect solution would require to run all the possible combinations of four steps. Inspired by Figure 3.4 and Figure 3.5, we narrowed the candidate thresholds of each step to four reasonable options. But, $4^4 = 256$ possible combinations are still intractable. Thus, we heuristically selected a few thresholds at each step for the succeeding steps.

As described above, 90% of **Non-REG** (TRIGGER,THEME) pairs have dependency path lengths smaller than five, so we tested thresholds from 2 to 5. Dependency path features are discarded when the dependency paths are longer than this threshold. Table 3.1 clearly shows the effect of discarding long dependency paths. All the experiments with thresholding get a better F_1-score. The best F_1-score was obtained when thresholds are 5. We remarked that increasing the dependency path length generally improves the precision but decreases the recall. However, experiments involving no threshold got the highest recall and lowest precision.

Table 3.2 illustrates the results using different thresholds in the *Binding* THEME fusion step. Unlike for the previous step, both recall and precision reach their highest value when threshold equals **5**. Comparison between different rows is more complex because of the dependency between steps. Let us separate the *Binding* events into two categories: single argument *Binding* events that contain one THEME and multi-argument *Binding* events that contain two THEMEs. The difference between experiments of *Binding* THEME fusion step are caused by two reasons: first is the difference of single argument *Binding* event listed above; second is the possibility of true/false recognition of multi-argument *Binding* event provided by (TRIGGER,THEME) pairs extracted in preceding step. In Table 3.3, we subtract the recall and precision of *Binding* event from preceding step in order to exclude single argument *Binding* event. We can see that the benefit of higher recall in preceding step is minor, which is smaller than 0.3% when difference of single argument is more than four. Contrary to recall, precision varies more significantly between experiments based on different preceding steps. Until now, we removed the configuration, which threshold equals three in **Non-REG** (TRIGGER,THEME) pair extraction, because its performances on *Binding* and **SVT** events are all uncompetitive.

TABLE 3.1: Investigating trimming of dependency path for **Non-REG** (TRIGGER,THEME) extraction step

Event Class \ Threshold		2	3	4	5	No
Gene_ expression	F_1-score	76.1	75.8	**77.1**	76.6	74.8
	recall	76.3	75.6	75.6	72.8	77.5
	prec.	75.9	76.0	78.7	80.8	72.2
Transcription	F_1-score	43.2	46.3	46.7	**50.3**	44.2
	recall	35.7	37.8	35.7	38.8	34.7
	prec.	54.7	59.7	67.3	71.7	60.2
Protein_ catabolism	F_1-score	**86.7**	78.7	82.8	81.4	82.0
	recall	86.7	80.0	80.0	80.0	83.3
	prec.	86.7	77.4	85.7	82.8	80.7
Phosphorylation	F_1-score	73.1	73.3	73.3	**74.3**	73.0
	recall	64.8	65.3	64.8	64.3	64.3
	prec.	83.9	83.4	84.5	87.9	84.4
Localization	F_1-score	71.6	**71.8**	70.7	71.1	71.6
	recall	61.4	61.4	59.4	59.9	61.9
	prec.	85.8	86.4	87.3	87.4	71.6
SVT	F_1-score	72.6	72.6	73.4	**73.4**	72.0
	recall	68.4	68.1	67.5	66.2	68.8
	prec.	77.5	77.7	80.4	82.5	75.4
Binding	F_1-score	**27.1**	25.4	23.9	24.8	26.6
	recall	31.4	28.4	27.1	26.5	30.3
	prec.	23.8	22.9	23.9	23.2	23.7

TABLE 3.2: Investigating trimming of dependency path for *Binding* THEME fusion step. Prev. refers to the thresholds used in previous step.

Prev. \ *Binding*		threshold = 4	threshold = 5	threshold = 6
threshold = 2	F_1-score	38.5	**41.0**	39.6
	recall	44.8	**47.7**	46.1
	prec.	33.7	**36.0**	34.8
threshold = 3	F_1-score	N/A	N/A	39.0
	recall	N/A	N/A	44.0
	prec.	N/A	N/A	35.0
threshold = 4	F_1-score	37.5	**40.9**	40.2
	recall	40.0	**43.2**	42.4
	prec.	35.4	**38.8**	38.2
threshold = 5	F_1-score	37.0	**40.3**	39.6
	recall	39.7	**42.9**	42.1
	prec.	34.6	**37.9**	37.3

TABLE 3.3: Investigating trimming of dependency path for *Binding* THEME fusion step for multiple argument events

Prev \ *Binding*		threshold = 4	threshold = 5	threshold = 6
threshold = 2	recall	13.4	**16.4**	14.7
	prec.	9.9	**12.1**	10.9
threshold = 3	recall	N/A	N/A	15.6
	prec.	N/A	N/A	12.0
threshold = 4	recall	12.9	16.1	15.3
	prec.	11.5	**14.9**	14.2
threshold = 5	recall	13.1	**16.4**	15.6
	prec.	11.3	14.7	14.1

For each $threshold \in \{2, 4, 5\}$ for the **Non-REG** (TRIGGER,THEME) step and for the best threshold in *Binding* THEME fusion step (**5**), we launched three experiments using $threshold \in \{2, 3, 4\}$ to define the optimal threshold for **REG** (TRIGGER,THEME) step. All the results are listed in Table 3.4. Similar to *Binding* event, the best threshold value (**3**) achieves both highest precision and recall. Comparing between corresponding content from three tables, we find that they have almost the same recall. It means that the difference of recall in **Non-REG** (TRIGGER,THEME) step in our experiment hardly change the recall in **REG** (TRIGGER,THEME) step. In contrary, the precisions of preceding steps and current step have positive correlation. But, overall, there is no significant difference between those three groups of experiments regarding the threshold selected in the preceding step.

The last step of our approach which aims to assign optional CAUSE argument to extracted **REG** events contains two groups of pairwise features that represent the relation within (TRIGGER, CAUSE) and (THEME,CAUSE) pairs. We fixed the threshold of (THEME,CAUSE) to $2 \leq threshold \leq 5$, which means dependency path less than two or longer than five will be discarded, and try $threshold \in \{3, 4\}$ for (TRIGGER,CAUSE). The F_1-score of experiments are given in Table 3.5, where the first row indicates the thresholds used in **Non-REG** (TRIGGER,ARGUMENT) step, the second row indicates the thresholds used for (TRIGGER,CAUSE) pairs in current step. We can see that, at this step, different threshold settings do not affect much the F_1-score. Moreover, since we can get the final total score of the whole system at this final step, we use it directly. Finally, we selected **4** as threshold for (TRIGGER, ARGUMENT) pair in this step. Then we can try different thresholds for (THEME,CAUSE) pairs by fixing threshold of (TRIGGER,CAUSE) to four. Results listed in Table 3.6 are consistent with the distribution of dependency path in Figure 3.5.

Finally, we get the best threshold configurations for each step, which is $threshold = 4$ in step 1, $threshold = 5$ in step 2, $threshold = 3$ in step 3, $threshold = 4$ for (TRIGGER,CAUSE) and $2 \leq threshold \leq 5$ for (THEME,CAUSE) in step 4.

From another perspective, discarding long dependency paths using threshold can be thought as a specific nonlinear projection of feature vectors. This method wisely selectes some inefficient examples, which is definitely a remedy of imperfect encoding of dependency path. A set of hyper-parameters was selected using experiments on development

TABLE 3.4: Investigating trimming of dependency path for **REG** (TRIGGER,THEME) step

(A) threshold = 2 in step 1, threshold = 5 in step 2

		2	3	4
Regulation	F_1-score	5.9	**7.0**	7.0
	recall	4.2	5.3	5.3
	prec.	9.6	10.5	10.3
Positive_regulation	F_1-score	13.9	**14.4**	14.1
	recall	10.9	11.5	10.9
	prec.	19.0	19.2	19.9
Negative_regulation	F_1-score	13.9	**14.4**	12.9
	recall	10.2	10.8	9.4
	prec.	22.0	21.9	20.3
REG	F_1-score	12.6	**13.2**	12.5
	recall	9.6	10.2	9.5
	prec.	18.5	18.6	18.4

(B) threshold = 4 in step 1, threshold = 5 in step 2

		2	3	4
Regulation	F_1-score	6.0	**7.1**	7.1
	recall	4.2	5.3	5.3
	prec.	10.1	11.0	10.8
Positive_regulation	F_1-score	14.0	**14.5**	14.2
	recall	10.9	11.5	10.9
	prec.	19.6	19.8	20.5
Negative_regulation	F_1-score	14.0	**14.5**	13.0
	recall	10.2	10.8	9.4
	prec.	22.4	22.4	20.8
REG	F_1-score	12.7	**13.3**	12.7
	recall	9.6	10.2	9.5
	prec.	19.1	19.2	19.0

(C) threshold = 5 in step 1, threshold = 5 in step 2

		2	3	4
Regulation	F_1-score	6.0	**7.2**	7.1
	recall	4.2	5.3	5.3
	prec.	11.2	11.1	10.9
Positive_regulation	F_1-score	14.2	**14.7**	14.4
	recall	11.0	11.6	11.0
	prec.	19.8	20.0	20.7
Negative_regulation	F_1-score	14.1	**14.6**	13.0
	recall	10.2	10.8	9.4
	prec.	22.8	22.8	21.1
REG	F_1-score	12.8	**13.5**	12.8
	recall	9.6	10.3	9.6
	prec.	19.3	19.5	19.3

TABLE 3.5: Investigating trimming of dependency path for (TRIGGER,CAUSE) in **REG** CAUSE assignment step. Threshold of dependency path between (THEME,CAUSE) belongs to $[2, 5]$.

Threshold in first step	threshold = 2		threshold = 4		threshold = 5	
(TRIGGER,CAUSE) threshold	3	4	3	4	3	4
Regulation	29.0	29.0	**29.8**	**29.8**	29.4	29.4
Positive_ regulation	37.8	37.8	38.0	38.0	38.4	**38.4**
Negative_ regulation	34.7	**34.9**	34.5	34.5	34.2	34.2
REG	35.5	35.6	35.7	35.7	35.7	**35.7**
TOTAL	49.8	49.9	49.7	**50.3**	49.5	49.5

TABLE 3.6: Investigating trimming of dependency path for (THEME,CAUSE) in **REG** CAUSE assignment step. Threshold of dependency path between (TRIGGER,CAUSE) is 4.

Threshold in first step	threshold = 2			threshold = 4			threshold = 5		
(THEME,CAUSE) threshold	$[1, 5]$	$[2, 5]$	$[3, 5]$	$[1, 5]$	$[2, 5]$	$[3, 5]$	$[1, 5]$	$[2, 5]$	$[3, 5]$
Regulation	28.6	29.0	26.3	29.8	29.8	27.1	29.4	29.4	26.7
Positive_ regulation	37.4	37.8	37.1	37.5	38.0	37.1	37.9	38.4	37.5
Negative_ regulation	35.0	34.9	32.8	34.5	34.5	32.6	34.2	34.2	32.4
REG	35.3	35.6	34.1	35.4	35.7	34.2	35.5	35.7	34.3
TOTAL	49.7	49.9	49.2	49.6	**50.3**	49.0	49.4	49.5	48.8

set. Those hyper-parameters are not guaranteed to be the optimal ones, but they define the configuration that was used in final test.

3.4.1.2 Knowledge Base, Coarse Tokenization and Window Size

Apart from the dependency path, there are still three features that require hyper-parameter selection. Fortunately, they are much simpler than dependency path, and we introduce them together in this section.

Window Size This hyper-parameter determines the range of the window of neighboring tokens used as feature for the current token. In our pairwise extraction steps, we have two sizes to define the windows around the head tokens of candidate trigger entities and around candidate arguments. We started from the configuration used in UCLEED system, that is, two tokens before and after head tokens of candidate trigger entities and one token before and after head tokens of candidate arguments. We searched around this configuration, called (2,1) for short, and tried (1,1), (2,2) and (3,3). Table 3.7 shows the results of experiments using different window size in **Non-REG** (TRIGGER,THEME) step, all the settings in other three steps remaining identical. Larger window sizes basically lead to higher precision for **SVT** events and *Binding* event, but lower performance on **REG** events. Our experiments confirmed that (2,1) is the best configuration for window size and we applied it to **REG** (TRIGGER,THEME) pair extraction as well.

TABLE 3.7: Investigating window sizes for **Non-REG** (TRIGGER,THEME) pair extraction. In (a, b), a denotes the window size for the candidate trigger entity, and b the window size for the candidate argument.

Event Class \ Window Size		(1,1)	(2,1)	(2,2)	(3,3)
Gene_ expression	F_1-score	74.9	**76.1**	74.7	75.7
	recall	75.3	**76.3**	73.6	71.6
	prec.	74.4	75.9	75.8	**80.3**
Transcription	F_1-score	41.0	43.2	43.0	**46.3**
	recall	32.7	**35.7**	34.7	34.7
	prec.	55.2	55.0	56.7	**69.4**
Protein_ catabolism	F_1-score	83.3	**86.7**	84.2	83.6
	recall	83.3	86.7	80.0	76.7
	prec.	83.3	86.7	88.9	**92.0**
Localization	F_1-score	72.2	71.6	**72.2**	68.3
	recall	**62.0**	61.4	61.4	55.3
	prec.	86.5	85.8	87.7	**89.3**
Phosphorylation	F_1-score	72.7	73.1	**73.8**	70.6
	recall	64.3	**64.8**	64.8	59.6
	prec.	83.8	83.9	85.6	**86.5**
SVT	F_1-score	71.2	**72.6**	72.0	71.7
	recall	67.5	**68.4**	66.6	63.6
	prec.	76.8	77.5	78.0	**82.2**
Binding	F_1-score	39.4	**41.0**	40.3	40.4
	recall	45.0	**47.7**	45.6	41.8
	prec.	35.0	36.0	36.1	**39.0**
Regulation	F_1-score	**30.0**	29.0	29.2	28.5
	recall	22.9	**22.9**	21.5	22.2
	prec.	43.3	39.6	**45.5**	39.9
Positive_ regulation	F_1-score	37.2	**37.8**	36.9	36.3
	recall	31.7	**32.7**	31.7	30.6
	prec.	**44.9**	44.9	44.7	44.5
Negative_ regulation	F_1-score	33.3	**34.9**	32.0	30.7
	recall	26.2	**27.0**	24.0	23.0
	prec.	45.7	**49.3**	48.0	46.0
REG	F_1-score	34.9	**35.6**	34.2	33.4
	recall	28.5	**29.3**	27.4	26.8
	prec.	44.9	45.3	**45.7**	44.2
EVENT ALL	F_1-score	49.0	**49.9**	49.0	48.1
	recall	44.1	**45.0**	43.2	41.4
	prec.	55.3	55.7	56.5	**57.6**

TABLE 3.8: Influence of the features derived from the IntAct knowledge base and from coarse tokenization

Event Class \ Feature		Original	Knowledge base	Coarse tokenization
Gene_ expression	F_1-score	76.8	**77.4**	75.6
	recall	75.0	74.1	75.6
	prec.	78.8	81.0	75.5
Transcription	F_1-score	46.1	45.8	**47.1**
	recall	35.7	35.7	37.8
	prec.	64.8	63.6	62.7
Protein_ catabolism	F_1-score	82.8	**84.2**	81.4
	recall	80.0	80.0	80.0
	prec.	85.7	88.9	82.8
Localization	F_1-score	67.5	67.7	**69.1**
	recall	55.3	54.3	57.9
	prec.	86.5	89.9	85.7
Phosphorylation	F_1-score	73.8	73.8	**73.9**
	recall	64.8	64.8	65.3
	prec.	85.6	85.6	85.1
SVT	F_1-score	72.4	**72.7**	**72.7**
	recall	65.9	65.3	67.0
	prec.	80.4	82.1	77.8
Binding	F_1-score	40.7	40.7	**41.8**
	recall	44.2	44.0	47.2
	prec.	37.8	37.9	37.5
Regulation	F_1-score	**29.4**	29.2	29.1
	recall	23.6	22.9	23.6
	prec.	39.0	40.4	37.9
Positive_ regulation	F_1-score	37.4	37.7	**37.8**
	recall	31.6	31.0	31.8
	prec.	45.8	48.2	46.7
Negative_ regulation	F_1-score	33.8	33.2	**36.2**
	recall	25.7	25.3	28.1
	prec.	49.5	48.4	50.9
REG	F_1-score	35.0	35.0	**35.9**
	recall	28.4	27.9	29.3
	prec.	45.6	47.0	46.4
EVENT ALL	F_1-score	49.4	49.6	**50.0**
	recall	43.4	42.9	44.6
	prec.	57.4	58.7	56.8

Knowledge Base The first two groups of results illustrated in Table 3.8 refer to experiments with or without features from IntAct knowledge base at every step. From this table, we can see that knowledge-base features slightly decrease recall and slightly increase precision of almost every event type. Overall, knowledge-base features do not have a significant influence on the final F_1-score.

Coarse Tokenization The third group of results in Table 3.8 belongs to experiment using features based on coarse tokenization. Compared to the original group, these features improved the recall of most event types but decreased the precision of **SVT** event. Overall, they bring a slightly higher improvement than knowledge-base features.

3.4.2 Test Results

In this section, we introduce our final results on BioNLP 2011/2013 test sets. These results are obtained from training on the training and development sets, in order to use a maximum number of examples in the training phase. We also merged the training sets of BioNLP 2011 and BioNLP 2013 for the results of the 2013 shared task.

Optimal features discovered in the previous section were used for both data-sets. We remind that (TRIGGER,THEME) pair extraction steps make use of features for candidate trigger entities, candidate arguments and pairs except features designed for two proteins arguments. *Binding* THEME fusion and **REG** CAUSE assignment steps do not use candidate trigger entity groups but use features for two proteins arguments. Candidate trigger entities and candidate arguments involve different range of neighbor tokens, that is, **2** before and after candidate trigger entities and **1** before and after candidate arguments. Dependency paths were discarded when their length l was (strictly) greater than 4 for **Non-REG** (TRIGGER,THEME) pair extraction, greater than 5 for *Binding* THEME fusion, greater than 3 for **REG** (TRIGGER,THEME) pair extraction, greater than 4, and greater than 5 or smaller than 2 for **REG** CAUSE assignment. Coarse tokenization based features and knowledge-base features were all included in feature set.

Note that we carefully designed a multi-class classifier, where hyper-parameters were selected by means of 10-fold cross-validation.

3.4.2.1 BioNLP 2011

Table 3.9 lists the results of our pairwise model along with the three most representative models from previous works. UCLEED [20] is the joint model that achieved the highest score in BioNLP 2011 shared task, SEARN [65] is a stacked model that extracts recursive events using old predictions and TEES [24] was the best pipeline model in BioNLP 2011. We can see that our pairwise model achieves the highest F_1-score on **SVT** and *Binding* events. This result strengthens our assumption about the drawbacks of typical pipeline models. Surprisingly, the joint model, which considers the relations altogether, is also slightly worse than our model. This could be explained by our well-designed features, and also because the joint model considers all the relations together, but most of them are not exist. Trying to learn all relationships requires more flexible models and accurate feature representations. However, UCLEED and SEARN systems got higher F_1-scores

TABLE 3.9: Results on the BioNLP 2011 test set

Event Class	UCLEED	SEARN	TEES	pairwise
SVT	73.5	71.8	72.1	**74.3**
BIND	48.8	45.8	43.3	**49.1**
REG	**43.8**	43.0	42.7	41.4
ALL	**55.2**	53.5	53.3	54.2

on **REG** events, which could be expected since our system cannot handle the recursive **REG** events appropriately.

3.4.2.2 BioNLP 2013

We participated the BioNLP 2013 Genia shared task. Therefore, more attention was paid to the BioNLP 2013 test set, but a bug caused a poor overall performance at the official challenge evaluation. After bug fix, our pairwise system actually achieves the best result among all officially evaluated results. Table 3.10 lists test results of our pairwise system, a counterpart pipeline system, and EVEX that is the winner of BioNLP 2013 Genia shared task. The counterpart pipeline system was derived from the pairwise system by replacing (TRIGGER,THEME) pair extraction by TRIGGER detection and edge detection. The features used in our pairwise system and its pipeline counterpart system are identical. They seem to be more appropriate than the ones of EVEX, since our pipeline counterpart wins by 3.3% on **SVT** events. The additional improvements of the pairwise classifier on the **SVT** events show the advantage of pairwise extraction.

Our systems are weak at detecting *Binding* events and **REG** events. While the lower **REG** performance might be due to the lack of potential to detect recursive events, we conjecture that the failure in *Binding* events is due to the *Binding* THEME fusion step. Table 3.11 displays the results of *Binding* (TRIGGER,THEME) pair detection and of the overall *Binding* events detection on BioNLP 2013 test set; BioSEM achieves the best *Binding* event score on this task. Comparing the *Binding* (TRIGGER,THEME) scores, our pairwise model gets the highest recall and the second F_1-score. Generally speaking, the lose of *Binding* events can be caused by two reasons: first, the *Binding* (TRIGGER, THEME) pairs are not well predicted; second, the extracted pairs are not correctly merged. The first condition can also be divided into two refined cases for multi-argument *Binding* events: all the pairs are not well predicted; one of the necessary pairs is not well predicted. As our approach gets higher pairwise extraction performance but lower multi-argument *Binding* event extraction performance, two possible explanations are: our approach always fails to extract one of the necessary pairs; our approach is weak on merging arguments. We believe that the second explanation is probably true because conditions in the first explanation are unlikely to happen frequently. A better evidence is that the BioSEM model achieves the highest *Binding* event recall with a lower *Binding* (TRIGGER,THEME) recall. Notice that BioSEM is a rule-based system that extracts patterns in predefined forms. It is a joint model that recursively extracts events by adding first the smallest and most frequent matching pattern. However, UCLEED, which also extracts *Binding* events jointly, got worse performance on BioNLP 2011 test set. It is hard to determine the crucial reason of BioSEM's success on *Binding* event.

TABLE 3.10: Results on the BioNLP 2013 test set

Event Class \ Model		Pairwise	Pipeline Counterpart	EVEX
Gene_ expression	F_1-score	**85.3**	83.0	82.7
	recall	80.9	76.4	81.4
	prec.	90.1	90.8	84.0
Transcription	F_1-score	**61.7**	60.4	55.0
	recall	65.4	57.4	54.5
	prec.	58.4	63.7	55.6
Protein_ catabolism	F_1-score	**62.5**	59.1	56.3
	recall	71.4	92.9	64.3
	prec.	55.6	43.3	50.0
Localization	F_1-score	60.2	60.6	**60.7**
	recall	50.5	44.4	47.5
	prec.	75.8	78.7	83.9
Phosphorylation	F_1-score	**79.9**	82.5	71.5
	recall	70.7	82.5	73.8
	prec.	75.8	82.5	69.4
SVT	F_1-score	**79.2**	77.8	74.5
	recall	74.4	72.5	71.6
	prec.	80.5	83.9	77.7
Binding	F_1-score	38.3	39.9	**42.9**
	recall	43.2	43.5	41.1
	prec.	34.4	36.9	44.8
Regulation	F_1-score	**28.1**	22.7	23.4
	recall	21.9	18.8	18.1
	prec.	39.1	28.9	33.3
Positive_ regulation	F_1-score	**40.2**	36.5	39.2
	recall	33.6	33.7	32.5
	prec.	50.1	39.9	49.3
Negative_ regulation	F_1-score	37.9	36.1	**43.9**
	recall	31.9	34.0	40.1
	prec.	46.5	38.4	48.4
REG	F_1-score	37.9	34.6	34.2
	recall	31.4	31.6	27.4
	prec.	47.7	38.2	**45.7**
EVENT ALL	F_1-score	**51.2**	48.0	51.0
	recall	46.0	44.8	45.4
	prec.	57.7	51.7	58.0

TABLE 3.11: *Binding* (TRIGGER,THEME) pair scores and full event scores on the BioNLP 2013 test set

		Pairwise	Pipeline Counterpart	EVEX	BioSEM
(TRIGGER,THEME) detection	F_1-score	61.7	61.4	59.0	**63.3**
	recall	**62.0**	59.8	56.5	57.7
	prec.	61.5	63.1	61.2	**70.1**
full event detection	F_1-score	38.3	39.9	42.9	**49.8**
	recall	43.2	43.5	41.1	**47.5**
	prec.	34.4	36.9	44.8	**52.3**

3.5 Conclusion

In this chapter, we introduced our pairwise model, which jointly extracts (TRIGGER,THEME) pairs. This model enables classifiers to use pairwise features to detect the TRIGGER, which achieves superior performances on single argument events and outperforms the total F_1-score of EVEX, the best system in BioNLP 2013 Genia shared task. Although pairwise predictors were already tested in previous works in BioNLP 2009, they lead to very bad performances, due to an ill-posed classification structure. Parts of these failures may also be due to the processing of multi-argument events and recursive events, which was not clearly explained. Our approach extracts **Non-REG** and **REG** (TRIGGER,THEME) pairs separately in order to partly detect the **REG** events using **Non-REG** events as arguments. Two additional steps are used to build multi-arguments *Binding* events and **REG** events after the extraction of (TRIGGER, THEME) pairs. The major drawback of our approach is that we cannot extract recursive **REG** events, that use other **REG** events as THEME arguments. To correct this drawback, one must extract **REG** events using a dynamic process. A straightforward solution with our current model would be run the last two steps in a cyclic manner. Considering the complexity of **REG** events, this plan would be very inefficient. In the next chapter, we present a model that completely solves the recursive problem in a very compact and efficient way. Deeper discussions about multi-class classifier and dependency features are developed in next chapter as well.

Chapter 4

Recursive Pairwise Model

In the previous chapter, we proposed a pairwise system that is in-between typical pipeline models and joint models. Our model reaches a good compromise between effectiveness and efficiency, with better performance than typical pipeline models as measured on Genia task, along with a much smaller computational complexity than joint models. However, the solution used to deal with recursive events is very rough, limited to possible relations between candidate entities and extracted **SVT** events. Compared to other works that participated the BioNLP Genia task, this shortcoming greatly reduces the advantage gained from pairwise extractions. In this chapter, we propose a RecUrsive Pairwise Event Extraction (RUPEE) system that contains a central (TRIGGER,THEME) pair extraction step, which is a dynamical process that incrementally assesses new candidate pairs. Our method yields the best results reported so far on the Genia event extraction task of the BioNLP 2011 and 2013 challenges.

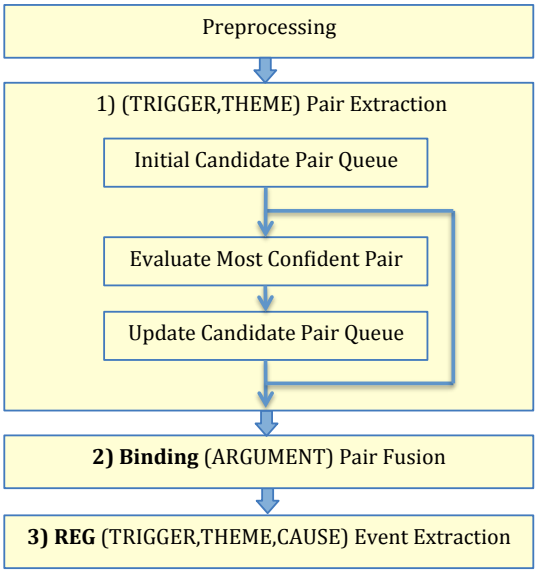


FIGURE 4.1: RecUrsive Pairwise Event Extraction (RUPEE)

4.1 Introduction

As described in Section 3.5, simply repeating the **REG** event extraction step would largely increase the complexity of the algorithm. To solve recursive events in the pair-wise framework without largely increasing the complexity, we developed RUPEE (RecUrisive Pairwise Event Extraction). RUPEE avoids the repetition of **REG** event extraction by extracting recursively the essential (TRIGGER,THEME) pairs in **REG** events. Applying the same idea to *Binding* event, the extraction of complex event is then separated into two parts: the extraction of essential pairs that determines the graph constituted by the main (TRIGGER,THEME) pairs in a sentence, and the complement of additional arguments that completes the full event extraction. Under this framework, **Non-REG** and **REG** (TRIGGER,THEME) pair extractions are merged in a single step, which is simple and efficient. We assume that recursive events constitute a directed acyclic graph, where the ancestor of a candidate entity cannot be used as its argument. Both (TRIGGER,THEME) pair extraction and **REG** CAUSE assignment steps are implemented in a constrained dynamic process that prevents the creation of cycles. *Binding* events do not have the same additional argument as **REG** events, one needs to merge two essential *Binding* pairs into a single event, but this process does not prevent discovering **REG** events nor does it generate unnecessary candidate pairs. We keep the *Binding* ARGUMENT fusion step of the pairwise system.

We simplified the classifier because the benefit brought by the post-processing of the SVM scores was not obvious in RUPEE. The new classifier only retains the SVMs in one-vs-rest framework. These hyper-parameters are set individually, for each SVM, so as to maximize the F_1-score of the corresponding event type taken in isolation. In the experimental results, the confusions between event classes are marginal: once an event trigger is detected, it is assigned the correct class in the vast majority of cases. Therefore, we believe that the shortcomings of the one-vs-rest framework are not responsible for the ones of our system.

We also investigated the detailed effect of the dependency path features (*E-walk* / *V-walk*).Intuitively, V-walks, which encode the dependency path into a more human readable triplet, contain more useful information. And it is easier for one to reconstruct the original dependency path from both V-walk and E-walk than just using one. Therefore, we tested two assumptions: first, V-walk features are more relevant to the relation extraction task, second, using all dependency path features is better than using only one of them. Experimental results illustrate unexpected phenomenon and provide believable evidence to overturn our assumptions made before. It also inspires us that combination of non-linear model and dense representation can be a promising future direction of encoding dependency paths.

4.2 Improved Recursive Classification Framework

The Chapter 3 showed the benefits of extracting pairs compared to pipeline models that separate trigger classification and edge detection in two steps. Our experiment confirms our assumption that putting off the prediction of event trigger brings essential information to extract potential triggers. Since our performance on **SVT** events is comparable to joint models, we believe that pair extraction is a good balance between joint global model and fine-grained decomposition. However, our pairwise model cannot capture potential event triggers for recursive events.

We first recall how previous approaches solve this problem. Pipeline systems like TEES, TEES 2.1, EVEX predict triggers before introducing any edges and assess all the possible pairs between candidate entities. A post-processing step is applied in the end to ensure that a non-protein ARGUMENT must be an event. Similarly, Sætre et al. [42] extract the entire event patterns based on the extracted triggers. They created 148 patterns for **REG** events, where each pattern is handled by an independent classifier. Patterns are specified with respect to the type and number of edges and arguments, for example, single-argument **REG** events using *Binding* events as THEME and single-argument **REG** events using *Gene_expression* events as THEME will be treated differently. Kilicoglu and Bergler [34] proposed another solution by extracting **REG** event in a recursive top-down fashion. Given a extracted **REG** trigger, when a potential sub-event argument is detected, they will try to determine the validity of the sub-event. This is a dynamic process, in which the most confident pairs are selected in each step. In the other hand, UCLEED solves recursive **REG** events by jointly extracting them all. In addition to observing all the possible pairs, this model requires heavy inference between candidate edges and candidate entities to solve the extraction task.

If we integrate the solution proposed by TEES, TEES 2.1, EVEX in our pair extraction, we have to observe all the possible directed pairs between candidate entities. This solution brings obviously much computational complexity, which is equivalent to the joint model without inference. Another solution extracts recursive events in a top-down manner. However, it will make pointless predictions, when the algorithm finds any sub-event that is not valid in the end. Finally, joint models like UCLEED consider a super-set of candidates compared to our pairwise model. Since we are looking for a method that enables recursive events extraction in pairwise models without high complexity, joint models cannot provide any help.

In this section, we will introduce an approach that recursively extracts essential parts of events by dynamically adding candidate pairs according to confidence of predictions.

Compared to pairwise system, this new method merges the (TRIGGER,THEME) pair extraction steps of **Non-REG** and **REG** events together, making the system more compact and elegant. This method not only conserves the beneficial pairwise relation extraction, but addresses the recursive problem efficiently with little additional complexity.

4.2.1 Recursive

As discussed above, there are two major categories of methods to solve the recursive events problem: static methods and dynamic methods. Pipeline models that predict all the possible edges between extracted triggers are statics method. The static approach of extracting recursive events cannot narrow the search range, so that a full exploration is inevitable. Dynamic methods contain two sub-types: models that take global relations into account like UCLEED; models that solves decomposed sub-problems recursively like Kilicoglu and Bergler [34]. Global dynamic models require a recursive inference on the states of all the possible observations, which makes it the most complex model reported so far. Therefore, we construct a dynamic model that locally handles recursive events.

We recall that the task is to assign a THEME to each event. Assigning a CAUSE to extracted **REG** events would not question the validity of previous extractions. Similar to CAUSE assignment, ARGUMENT fusion for *Binding* events cannot question the validity of extracted *Binding* (TRIGGER,THEME) pairs. All the changes brought by these two steps can be dealt with post-processing. As both steps maintain extracted (TRIGGER,THEME) pairs, (TRIGGER,THEME) pair extraction allows to determine the existence of events. There is no need to match various event patterns in a dynamic process. Hence, dynamically extracting (TRIGGER,THEME) pairs is the most efficient way to solve recursive events extraction problem in our task.

Top-down and bottom-up are two strategies of information processing and knowledge representation. A top-down approach essentially breaks down a complex system to gain insight into its sub-systems. Oppositely, a bottom-up approach puts together sub-systems to model more complex systems. Compared to top-down approach, bottom-up processing is the way humans build concepts from incoming sentences. We selected the bottom-up approach to extract the recursive events in our model.

Algorithm 4 presents the dynamic recursive process that resolved recursive events with small additional complexity. Starting from extract list of a candidate pair initialized by all the pairs between candidate entities and proteins, we select each step the prediction with the highest score. Each time a pair $p_{\alpha\beta}$ is predicted as an event, the algorithm adds new candidate pairs $\{(t_i, a_\alpha)\}$, where $p_{\alpha\beta}$ refers to a pair of trigger t_α and argument a_β. Notice that multiple events could share the same candidate trigger t_α, which leads the algorithm to add dynamically some already detected pairs into candidate set. Thus, we logged all the detected pairs to ensure the termination of algorithm. In spite of detected pairs, we also remove the pairs that might form circles with respect to detected events because we assume that recursive events constitute a directed acyclic graph, where the ancestor of a candidate entity cannot be used as its argument. The loop ends when there is no undetected candidate pair left in a sentence.

Algorithm 4 Extracting events with RUPEE

input sentence S, candidate entities $\mathcal{T}_S = \{t_\alpha\}_\alpha$ and candidate argument $\mathcal{A}_S = \{a_\beta\}_\beta$

initialize candidate pairs $\mathcal{P}_S = \{(t_\alpha, a_\beta),\ t_\alpha \in \mathcal{T}_S,\ a_j \in \mathcal{A}_S\}$
initialize detected pairs $\mathcal{D}_S = \emptyset$
initialize extracted events $\mathcal{E}_S = \emptyset$
score and label the pairs in $\mathcal{P}_S$
while $\mathcal{P}_S \neq \emptyset$ **do**
  select the pair $p_{\alpha\beta} \in \mathcal{P}_S$ with highest score
  update $\mathcal{P}_S \leftarrow \mathcal{P}_S - \{p_{\alpha\beta}\}$
  update $\mathcal{D}_S \leftarrow \mathcal{D}_S \cup \{p_{\alpha\beta}\}$
  if $\hat{y}_{\alpha\beta} \neq$ None **then**
    create event $\hat{e}_{\alpha\beta} = (t_\alpha, a_\beta, \hat{y}_{\alpha\beta})$
    update $\mathcal{E}_S \leftarrow \mathcal{E}_S \cup \{\hat{e}_{\alpha\beta}\}$
    update $\mathcal{P}_S \leftarrow \mathcal{P}_S \cup \{(t_i, t_\alpha) | t_i \in \mathcal{T}_S\}$
    prune $\mathcal{P}_S \leftarrow \mathcal{P}_S - \mathcal{D}_S$
    censor pairs $\mathcal{P}_S$ to avoid cycles with respect to $\mathcal{E}_S$
    score and label the new $\{(t_i, t_\alpha)\}$ pairs
  end if
end while
return extracted events $\mathcal{E}_S$

4.2.2 Merging the Trigger-Theme Steps

The previous pairwise model separates (TRIGGER,THEME) predictions of **Non-REG** and **REG** events into two steps for the purpose of using **Non-REG** events as candidate arguments in **REG** events extraction. Since the recursive algorithm introduced above dynamically adds candidate pairs p between candidate entities t, we no longer need to split the prediction of **Non-REG** and **REG** (TRIGGER,THEME) pairs in RUPEE model. An arguable aspect is using different dependency features for **Non-REG** and **REG** events. Indeed, it enables us to choose different features for those two categories of events. For example, we used the dependency path features with different thresholds in **Non-REG** and **REG** (TRIGGER,THEME) pairwise extraction steps. From Figure 3.4, we can see that there is no remarkable difference between the distributions of **Non-REG** and **REG** events. In fact, the difference between *Positive_ regulation* and *Gene_ expression* is not larger than the difference between *Binding* and *Gene_ expression*. Though the evaluation results on BioNLP 2013 development set in Table 3.4 shown that **REG** pair extraction prefers 3 whereas **Non-REG** prefers 4, recursive pairwise extraction could have different performance because it faces different examples. Moreover, the optimal threshold of middle step in pairwise system does not necessary lead to the best final result of the complex event extraction system. Compared to the possible profit of using different thresholds, we prefer to arrange all the pair extractions in a uniformed framework. Note that it is possible to build a complex system, where each classifier (SVM) uses different features for the same example pair. However, Van Landeghem et al. [54] that used different feature sets for each fine-grained class, did not achieve competitive final results. Finally, we created classifiers to solve all the (TRIGGER,THEME) pairs in an unified framework using the same features.

The unified framework uses one multi-class classifier to predict candidate pairs among the nine event types. Creating the training sets for the unified classifier is more complicated: since they can take events as arguments, new pairs are added to $\mathcal{P}_S$ by considering all the

events already detected, as sketched in Algorithm 4. Hence, the set of training examples is not known before training, since it depends on predictions of classifier. Training with these pairs requires to use either online algorithms or complex search-based structured prediction procedures, such as in Vlachos and Craven [65]. In this thesis, we used instead a super-set $\mathcal{P}_S^{sup}$ that includes all the possible pairs. Notice that even with a perfect classifier that always makes correct predictions, our algorithm could generate different examples according to the evaluation order. Thinking about a sentence that contains three events $e_1 = (T1, P1)$, $e_2 = (T2, P2)$ and $e_3 = (T2, e_1)$ where $\{T1, T2\}$ are triggers and $\{P1, P2\}$ are proteins. Suppose there are only two candidate entities $t_1 = T1$ and $t_2 = T2$, evaluating candidate pair (t_2, t_1) before $(t_2, P2)$ will mask (t_1, t_2). Therefore, we used the super-set that contains all the possible pairs the classifier will encounter no matter what evaluation order is selected. The enriched training set is

$$\mathcal{P}_S = \{p_{ij} = (t_i, a_j) | t_i \in \mathcal{T}_S,\ a_j \in \mathcal{A}_S\} \cup \{p_{i\alpha} = (t_i, e_{\alpha\beta}) | t_i \in \mathcal{T}_S, \exists \beta : \hat{y}_{\alpha\beta} \neq None\}$$

In addition to the uncertainty problem brought by evaluation order, we still have to specify the label $\hat{y}_{\alpha\beta}$ used in the above formula. We simply use the true label $y_{\alpha\beta}$ as alternative. This allows to define all training examples beforehand and hence to use standard batch SVM algorithms. The drawback is that, since extracted events in test are not necessarily correct, using true labels during trainign creates a divergence between training and testing scenarios, which can lead to degraded performance. However, as our experiments show, this effect is marginal compared to the advantages of using fast reliable batch training algorithms for SVMs.

4.2.3 Complexity

Though being rather minimalist, the pairwise structure captures a great deal of interactions, and the simplicity of the structure leads to a straightforward inference procedure. Compared to pipeline models, it only has a slight increase in computational complexity. We denote $m = \text{card}(\mathcal{T}_S)$, the number of candidate entities, $n = \text{card}(\mathcal{A}_S)$, the number of annotated proteins and m' the number of detected triggers. The complexity of a pipeline model is $\mathcal{O}\big(m'(n+m')\big)$, whereas that of RUPEE is $\mathcal{O}\big(m(n+m')\big)$. This implies more calls to the classifying mechanism, but this is not too penalizing, since SVM-based classification scales well with the number of examples. Besides, this is still cheaper than joint models such as the one introduced in Riedel and McCallum [19], whose complexity is $\mathcal{O}\big(Rm(n^2+m)\big)$, with R the number of iterations of the inference solver (it seems that $\max(R) \approx 10$). As shown in Section 4.6, our system, with moderate complexity, outperforms pipeline models and is competitive with joint ones.

4.3 Implementation

In the previous model, we used classifier with post-processing on SVM scores, which show better performance than simple SVMs on BioNLP 2013 development set. Nevertheless, the post-processing on SVM scores shown instable performance when evaluating under the new recursive framework. Therefore, SVMs in one-vs-rest framework, was finally selected for the recursive model. More experiments of different classifier variants are in Chapter 5.

In Section 3.3.2.2, we discussed the effectiveness of *E-walk*/ *V-walk* encoding in relation extraction. Follow the discussion; we selected the valuable dependency path by trimming at each step. The results proved the success of this strategy. In this section, we pursue the discussion about the methods encoding the dependency path. Though *E-walk* and *V-walk* encoding methods are very similar, they give different view of the dependency path. We looked into the implementation detail of *E-walk* and *V-walk* and ran experiments with only one of them.

4.3.1 Simplified Classifier

In Section 3.3.1, we used SVMs in one-vs-rest settings to solve the multi-class classification problem. Compared to the standard one-vs-rest framework, the major difference we made is selecting different hyper-parameters for each SVM via cross-validation. Denote that we selected hyper-parameters that achieve the highest *potential* F_1-score computed via a list of thresholds. This solution was initially designed hope to get better result via threshold tuning in later steps. Even for the simplified classifier, this method brings slightly better result than selecting hyper-parameter with respect to optimal F_1-score without thresholds on BioNLP 2013 test set. Hence, we kept this in our classifier.

Under the one-vs-rest framework, the classifier corresponding to a class was trained using examples belonging to all the other classes as negative examples. Consequently, the positive and negative classes are unbalanced. It is much worse in our task, because we now increase amount of examples in *None* type. In the trigger detection step in typical pipeline models, the largest proportion of positive examples to negative example will be less than $1:10$. This number becomes less than $1:100$ in (TRIGGER,THEME) pair extraction step for RUPEE. Due to the highly unbalanced data sets, we used C^+/C^- that specify different costs for positive and negative examples. The final decision rule simply consists in predicting the class corresponding to the highest SVM score.

4.3.2 Edge-Walk vs Vertex-Walk

As mentioned before, we argued the effectiveness of representing dependency path by triplet encoding (*E-walk*/ *V-walk*). Figure 2.5 illustrates the triplet formatting: from the dependency parse given on top, three V-walk triplets and two E-walk triplets are defined. Those triplets are further encoded into tuples (*dep-tag*, *token*, *dep-tag*) for E-walk; and (*token*, *dep-tag*, *token*) for V-walk. In these tuples, *tokens* are described by stemmed token and POS tags, and *dep-tags* are dependency labels. Note that we use an indicator *PROT* instead of stemmed token when *token* belongs to a protein. In order to increase the generalizability of feature, we created new features, which are tuples that only contain pairs of the triplets.

The dependency path is a sequence constituted by staggered edge nodes and vertex nodes. Before being inserted into feature vector as a bag-of-triplets, *E/V-walk* can be treated as two triplet sequences that cover the dependency path by a sequence of overlapping triplets, where the first element of the jth triplet is the last element of the $j-1$th triplet. The only difference between them is their starting node that *V-walk*s start at token, *E-walk*s start at dependency tag. Ignoring the difference between dependency tags and words, both walks contain the equivalent information to reconstruct the original dependency path. Nevertheless, the bag-of-triplets process breaks the sequence and

linear classifier does not aware of correlations between features. The quality of *E/V-walk* features could be discussed via single triplet. Triplets in *V-walk* sometimes directly solve the event extraction problem, such as *regulation,PREP_ of,promoter* has very similar form to the task definition. As triplets of *V-walk* contain two tokens, whereas triplets of *E-walk* contain only one token, *V-walk* brings obviously more information than *E-walk* according to the information entropy theory. Therefore, we made assumption that *V-walk* features are more important than *E-walk* feature. Thus, we run experiments to measure their importance by the loss incurred by their withdrawal from the model. See Section 4.4.2 to get the detailed results.

4.4 Experiments

To demonstrate the performance and efficiency of RUPEE, we ran experiments on the BioNLP 2011 and 2013 data-sets. We also ran UCLEED on the BioNLP 2011 data-sets to compare training times. Besides, we ran experiments to study the relevance of some choices related to classifiers or features.

4.4.1 Classifiers

Compared to the previous pairwise model, we simplified the classifier in RUPEE. In this classifier, we tune the hyper-parameter (C^+/C^-) for each SVM individually to optimize the potential F_1-score of corresponding class. In order to prove the effectiveness of this classifier, we also ran experiments with SVMs using identical hyper-parameter and SVMs optimize F_1-score with thresholds equal 0. We tested the four classifiers under RUPEE with three data-set configurations listed in Table 4.1.

TABLE 4.1: Data-set configurations used to test different classifiers.

	Training set	Test set
1)	BioNLP11 train&BioNLP11 development	BioNLP11 test
2)	BioNLP11 train	BioNLP11 development
3)	BioNLP11 train&BioNLP11 development BioNLP13 train&BioNLP13 development	BioNLP13 test

All the features and cross-validation settings are identical to the ones presented in the previous chapter.

TABLE 4.2: Total F_1-scores for classifiers using different classifiers with different data-set configurations.

	BioNLP11 test	BioNLP11 devel	BioNLP13 test
Same SVMs	55.1	52.5	53.9
Diff SVMs	-/-	-/-	54.3
Potential Diff	55.7	52.8	**54.4**
Combination	**56.0**	**53.3**	54.2

Table 4.2 lists the total F_1-scores of four classifiers. All the four classifiers do not have significant differences on different data-sets. The SVMs with combination of scores, which

was selected in our pairwise model, returned the best performances on the BioNLP 2011 test set and development set. However, with limited number of trials on the BioNLP 2013 test set using the test server, we found that the advantage of the SVMs with combination of scores is not systematic. Consider the proportion of abstracts and full articles in these data-sets listed in Table 1.2, the data distributions of BioNLP 2011 development set and test set are closer to the training set than the BioNLP 2013 test set, which only contains full articles. Hence, we think that the better performances of SVMs with combination of scores on BioNLP 2011 development and test sets are caused by better fitting the training set. But it leads to worse performance on the BioNLP 2013 test set. Based on these limited trials on the test set, we selected the classifier that achieves the best total F_1-score on the BioNLP 2013 test set. Table 4.3 lists the detailed results of the four classifiers returned by the test server of the BioNLP 2013. Generally speaking, most of the F_1-scores are very similar between every classifier, where the classifier achieves highest recall on one class usually achieves the lowest precision on the same class. The F_1-scores only have big differences on *Transcription* events and *Protein_ catabolism* events. *Protein_ catabolism* class is a minor class that contains less than 50 examples, of which result can be highly variable. However, it is surprising that the performances on *Transcription* events are so different. The efforts made on classifiers are devoted to improve the final performance, but their improvements are not significant.

Comparing the simplest SVMs with identical hyper-parameter and the most complex SVMs with post-processing, we find that these two classifiers achieve the best recall and best precision respectively. Other two classifiers are compromised solutions between the two extreme situations. Except for SVMs using identical hyper-parameter, all other three classifiers have similar performances. The first and third rows of Table 4.2 show that the improvements from SVMs with same hyper-parameters to SVMs with different hyper-parameters are systematic. Consider the computing cost and final F_1-score, we selected SVMs using different hyper-parameter to optimize potential F_1-score in our system.

It is surprising to see the post-processing with SVM scores failing to achieve a better result. Post-processing of SVM scores was proposed to smooth and ameliorate the decisions made by the binary SVMs that are trained individually. Basically, it is designed to solve the conflicts between decisions in the multi-class classification problem. We suspected that the failure of SVMs with post-processing is caused by the specific distribution of examples in each event type. Confusion matrix, also known as contingency table, allows the visualization of the performance of the classifier. Given a classifier and a set of examples to evaluate, confusion matrix **CM** summarizes the number of examples with respect to their true class and predicted class.

$$\mathbf{CM}[i,j] = |\{n|\hat{y}_n = i, y_n = j\}|$$

We computed the confusion matrix on training and development sets used in BioNLP 2011 and 2013 via cross-validation. During the cross-validation, one confusion matrix is computed using a subset as test set and classifier trained on the other subsets. Finally, the global confusion matrix is created by adding all the confusion matrices computed above. Note that we did not compute the confusion matrix on test set because it is not accessible. Table 4.4 shows that the vast majority of errors are due to the detection of an event when there is none or to the absence of detection of an existing event: when an event is detected, its correct type is predominantly predicted.

TABLE 4.3: F_1-scores on the test set of the BioNLP 2013 GE task using different classifiers. *Same SVMs* refers to SVMs use the same hyper-parameter, *Diff SVMs* refers to SVMs use different hyper-parameters, *Potential Diff SVMs* refers to SVMs use different hyper-parameters that optimize potential F_1-score.

Event Class \ Classifiers		Same SVMs	Diff SVMs	Potential Diff SVMs	Combination
Gene_ expression	F_1-score	84.2	85.0	**85.1**	84.6
	recall	78.5	81.1	80.5	**81.4**
	prec.	**90.8**	89.5	90.4	88.1
Transcription	F_1-score	**68.2**	65.2	62.8	61.5
	recall	58.4	59.4	**64.4**	59.4
	prec.	**81.9**	72.3	61.3	63.8
Protein_ catabolism	F_1-score	64.5	64.5	68.8	**72.7**
	recall	71.4	71.4	78.6	**85.7**
	prec.	58.8	58.8	61.1	**63.2**
Localization	F_1-score	56.8	56.0	**57.7**	57.5
	recall	42.4	42.4	43.4	**44.4**
	prec.	85.7	82.4	86.0	**81.5**
Phosphorylation	F_1-score	81.4	81.4	**81.8**	81.5
	recall	79.4	80.6	81.3	**82.2**
	prec.	**83.6**	82.2	82.3	81.8
SVT	F_1-score	78.3	**78.5**	78.3	78.0
	recall	70.7	72.6	73.0	**73.2**
	prec.	**87.8**	85.5	84.6	83.5
Binding	F_1-score	42.2	**42.9**	42.4	42.4
	recall	39.9	43.2	41.4	**43.8**
	prec.	**44.6**	42.5	43.4	41.0
Regulation	F_1-score	30.2	31.2	**31.8**	31.7
	recall	25.4	28.1	30.6	**28.8**
	prec.	**37.2**	35.2	33.1	35.2
Positive_ regulation	F_1-score	44.7	46.1	**46.3**	**46.3**
	recall	37.4	40.5	39.7	**41.3**
	prec.	**55.5**	53.4	55.5	52.7
Negative_ regulation	F_1-score	**43.8**	43.6	43.6	43.5
	recall	36.9	39.0	38.0	**39.2**
	prec.	**53.8**	49.5	51.0	48.8
REG	F_1-score	42.3	43.2	43.2	**43.3**
	recall	35.4	38.3	37.9	**38.9**
	prec.	**52.3**	49.5	50.2	48.9
EVENT ALL	F_1-score	53.9	54.3	**54.4**	54.2
	recall	46.8	49.4	49.1	**50.1**
	prec.	**63.4**	60.2	60.8	59.0

TABLE 4.4: Confusion matrix for RUPEE on the BioNLP 2013 GE task, computed by cross-validation on the training and development sets

	Prediction										
Truth	None	Gene_exp	Trans	Pro_cat	Phosp	Local	Bind	Regul	Pos_reg	Neg_reg	
None	223460	404	163	27	42	60	296	390	799	397	226038
Gene_exp	440	2741	13	0	0	17	2	1	43	5	3262
Trans	186	16	565	0	0	0	0	4	15	0	786
Pro_cat	30	0	0	150	0	0	0	0	1	0	181
Phosph	76	0	0	0	413	0	0	0	0	0	489
Local	114	20	0	0	0	398	4	0	1	2	539
Bind	507	0	0	0	0	1	1470	2	0	1	1981
Regul	453	0	0	0	0	0	1	813	33	4	1304
Pos_reg	1245	42	10	0	0	0	2	67	2456	7	3829
Neg_reg	555	7	2	1	1	0	0	46	11	1176	1799
	227066	3230	753	178	456	476	1775	1323	3359	1592	

We recall that we used a single hyper-parameter C to train the logistic regression classifiers. Because the logistic regression classifiers use output scores from SVMs as input data, their feature space is very small compared to the total number of training examples. Hence, we ignored the hyper-parameter problem, which is used to control the regularization. It is possible to get a better result if we optimize logistic regression classifiers in the same way as SVMs, that is, using asymmetric costs instead of C and setting different hyper-parameters for each class.

4.4.2 Features

We discussed about the quality of different encodings for the features of the dependency path in Section 4.3.2. A hypotheses was made that *V-walk* features are more important than *E-walk* features. As the final score represents the performance of final events that involves the *Binding* argument fusion and **REG** CAUSE assignment, measuring the benefits of joint feature via final score will be difficult due to the complexity of system. Hence we focused on the scores for decomposed (TRIGGER,THEME) pairs, which directly reflect the performance of recursive pairwise step. Besides, we also want to eliminate the influence of calibration of classifier between precision and recall. Therefore, we drawn the precision-recall curve by manually setting the threshold used by the SVM corresponding to the *None* class. Notice that, changing this threshold cannot make the algorithm reach the limit of precision or recall, because there are still minor false predictions between different event types and over prediction made by our algorithm when eliminating correct pairs to prevent loop. All the scores were obtained by training classifier on training set from BioNLP 2011 and 2013 and evaluating on the development set from BioNLP 2013.

Figure 4.2 displays four precision-recall curves obtained with all the features, without *V-walk* features, without *E-walk* features and without all the dependency path features. Without all the joint features, we observed a huge drop in performance, combined with

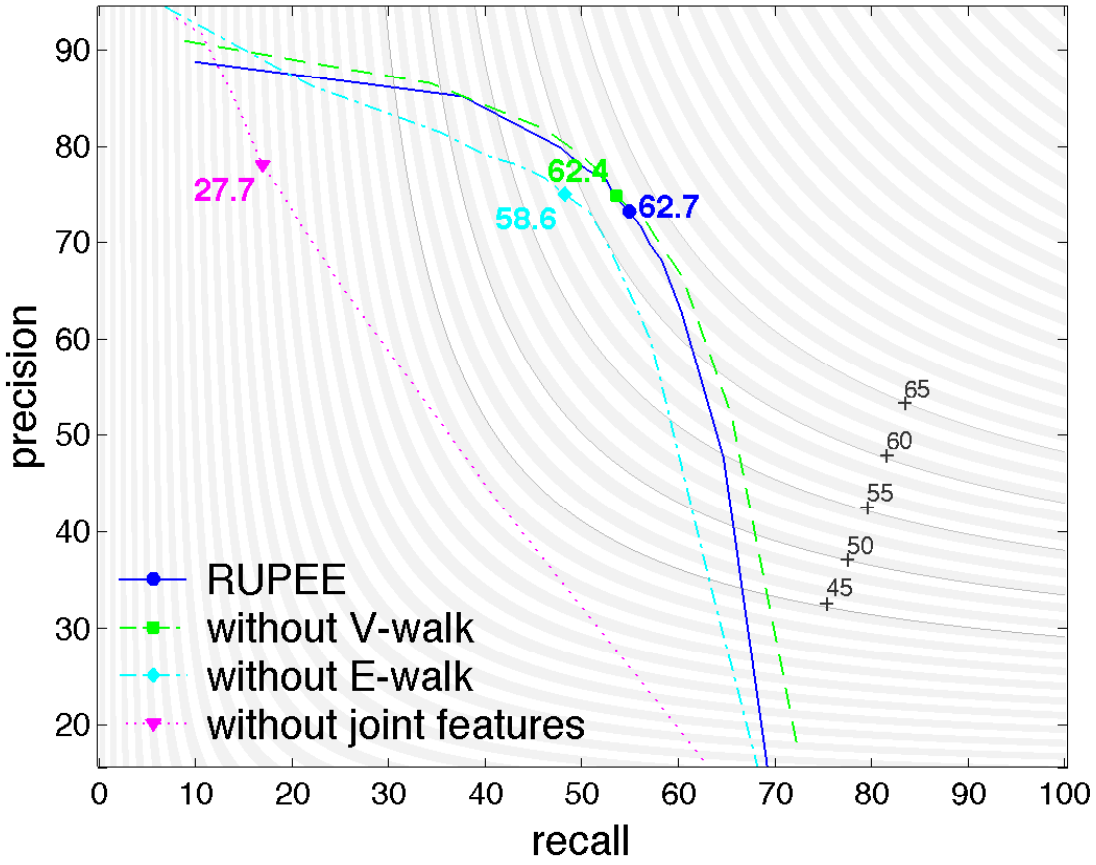


FIGURE 4.2: Precision-recall curve for (TRIGGER, THEME) classification, with or without joint features, on the BioNLP 2013 development set. Level curves of F_1-score are shown in the background.

a very bad calibration of the classifier that achieves 27.7 F_1-score. That means joint features are crucial to the relationship resolution. Without *E-walk*, we observed a significant drop in performance, which is 58.6 with calibration and will not exceed 60 when being better calibrated. Without *V-walk*, the performance is nearly the same as using all the features. We believe that *E-walk* plays a major beneficial role among the joint features, whereas *V-walk* has a marginal effect. This is opposite to what we supposed before.

A possible explanation is that *V-walk* is too sparse to be observed by a classifier in our task. Let us look at the feature dimensions of *E/V-walk* in training set we used here: *V-walk* contains 87139 features whereas *E-walk* contains 54030 features. These numbers involve the features using POS and stemmed tokens in place of tokens in triplets. Since the number of POS is fixed while number of tokens increases with the size of corpus, it is better to compare the dimensions of *E/V-walk* with only stemmed token, which is 81874 for *V-walk* and 42059 for *E-walk*. Similar to projecting vectors from low dimension space to high dimension space, *V-walk* is theoretically more expressive caused by the larger feature dimension. Though, we made the assumption that *V-walk* is less effective because its sparsity, it is necessary to study the variation of the data matrix with respect to the class labels. Due to the time limit of my PhD, we did not finish this job.

TABLE 4.5: F_1-scores on the test set of the BioNLP 2011 GE task

Event Class	UCLEED	SEARN	TEES	Pipeline counterpart	RUPEE	pairwise
SVT	73.5	71.8	72.1	71.8	74.0	**74.3**
BIND	48.8	45.8	43.3	40.0	**50.5**	49.1
REG	43.8	43.0	42.7	35.7	**45.1**	41.4
ALL	55.2	53.5	53.3	50.0	**55.6**	54.2

4.4.3 Model Comparison

In this section, we evaluate empirically our system in the framework (data, annotations and evaluation) of biomedical event extraction defined in the GE tasks of the BioNLP challenges. More precisely, we present results on the test sets of the fresh 2013 GE task, and of the 2011 edition to compare to joint methods. As in Chapter 3, we trained our model on the merged sets that contain documents from training and development sets. And we also merged BioNLP 2011 and 2013 training and development sets to train the classifiers that participate in the 2013 shared task.

In order to assess the efficiency of our modeling choices, we also implemented a pipeline counterpart system, following the structure of the TEES approach [23–25] but using the same feature set, preprocessing and a similar post-processing as our system. This pipeline system comprises four steps:

1. TRIGGER classification, which assigns event types from $\mathcal{Y}$ to candidate entities $t_i \in \mathcal{C}_S$ using the same multi-class SVMs classifier as RUPEE.
2. EDGE detection, which identifies the edges between extracted triggers and proteins and between REG triggers and all the triggers; labels from $\mathcal{Y}_{edge} = \{\text{THEME}, \text{CAUSE}, None\}$ are assigned to those pairs using multi-class SVMs classifier as step 1.
3. *Binding* THEME fusion, identical to as in Section 3.2.2.1.
4. THEME-CAUSE fusion, given two predicted pairs $(t_i, \text{THEME} : a_\beta)$, $(t_i, \text{CAUSE} : a_\gamma)$, this step decides whether they should be merged into a single event $(t_i, \text{THEME} : a_\beta, \text{CAUSE} : a_\gamma)$.

We used features similar to the ones used in the pairwise model for RUPEE and its pipeline counterpart. Remind that the pairwise model uses different thresholds in first and third steps. Since RUPEE fuses these two steps together, we used threshold $l \leq 4$ in recursive (TRIGGER,THEME) pair extraction step. Similarly, we used the same threshold in edge detection step in the pipeline counterpart. All the other features are exactly the same as what we used in Chapter 3.

4.4.3.1 BioNLP 2011

Table 4.5 lists the results of RUPEE, its pipeline counterpart, our pairwise model introduced in the previous chapter, and those of UCLEED [19] and TEES [24], which are respectively the best performing joint model and best pipeline on this task. We also

TABLE 4.6: F_1-scores on the test set of the BioNLP 2013 GE task

Event Type or Class	TEES 2.1	EVEX	Pipeline counterpart	RUPEE	pairwise
Gene_ expression	82.7	82.7	83.9	85.1	**85.3**
Transcription	55.0	55.0	61.7	**62.8**	61.7
Protein_ catabolism	56.3	56.3	66.7	**68.8**	62.5
Phosphorylation	72.6	71.5	**81.8**	**81.8**	79.9
Localization	**63.3**	60.7	56.9	57.7	60.2
SVT	74.9	74.5	79.0	**79.6**	79.2
BIND	**43.3**	42.9	41.6	42.4	38.3
Regulation	23.0	23.4	23.1	**31.8**	28.1
Positive_ regulation	38.7	39.2	36.5	**46.3**	40.2
Negative_ regulation	43.7	**43.9**	38.1	43.6	37.9
REG	38.1	38.4	35.1	**43.2**	37.9
ALL TOTAL	50.7	51.0	50.8	**54.4**	51.2

added SEARN [65], which is a hybrid between them. The results for UCLEED, TEES and SEARN models are reproduced from [19, 24, 65] respectively.

In the previous chapter, we did not give a counterpart model for 2011 data, so that the benefit of pairwise extraction on 2011 data is not clear. Thought the counterpart model in this chapter is not designed based on pairwise model, we can still compare them to judge the benefits of feature and model structure. Both pairwise and RUPEE model got much higher scores than the pipeline counterpart on **Non-REG** events, which shows the advantage of pairwise extraction. Moreover, the pairwise model outperforms pipeline counterpart on **REG** events, which can extract recursive events. It might be because the poor (TRIGGER,ARGUMENT) pair performance of counterpart model largely hurt the final performance. Comparing the pairwise model and RUPEE, the recursive process got large improvement on **REG** events over the static model.

Compared to other models, the benefits of the pairwise structure and the recursive process are very obvious, thereby outperforming the overall F_1-score of all the selected models in this section. To the best of our knowledge, our model thus reaches the best overall performance reported so far on this data set for a single model. We do not compare with the results of FAUST [21], which achieved the best F_1-score on this task (56.0) because this is an ensemble of various models of UCLEED and of the Stanford system [17], which makes it an unfair comparison. By combining the use of the simple pair structure between triggers and arguments with a recursive prediction process, our approach is able to outperform pipeline models and to be at least at par with models relying on much more sophisticated structures. For this task, it is thus highly beneficial to consider pairwise interactions from beginning to end, but more complex dependencies seem not to be essential, especially since they come at a higher computational cost.

4.4.3.2 BioNLP 2013

Table 4.6 lists the detailed test F_1-scores, as returned by the official challenge test server (using the default *approximate span and recursive matching* evaluation setting). We

TABLE 4.7: *Binding* (TRIGGER,THEME) pair scores and event scores on BioNLP 2013 test set

		TEES 2.1	EVEX	Counterpart	RUPEE	pairwise	BioSEM
Binding-THEME pair extraction	recall	57.9	56.9	54.6	56.7	**62.0**	57.7
	prec.	60.5	61.2	69.5	68.5	61.5	**70.1**
	F_1-score	59.2	59.0	61.1	62.0	61.7	**63.3**
Binding full event	recall	42.3	41.1	39.6	41.4	41.7	**47.5**
	prec.	44.3	44.8	43.7	43.4	33.7	**52.3**
	F_1-score	43.3	42.9	41.6	42.4	37.3	**49.8**

compare our model to the winner of the challenge, EVEX [61], and of the best runner-up, TEES 2.1 [25], which are both pipeline approaches.

Our approach is slightly below TEES 2.1 on **BIND** events, but overall, it outperforms all competitors significantly (by more than 3%), with a wide margin on REG events. Different from the results on BioNLP 2011 test set, our pipeline counterpart has an overall performance similar to EVEX and TEES 2.1, while being better for SVT and worse for BIN and REG events. These disparities are due to the differences in features and in processing details. Consider that we selected the thresholds of dependency path features on the development set of BioNLP 2013, our features are more powerful on 2013 test set than 2011 test set. The benefits of the pairwise structure and the recursive process are demonstrated by the considerable improvement upon the pipeline counterpart (using the same features, pre- and post- processing). In particular, the recursive prediction process run on REG events brings about very substantial improvement (more than 5% over the pairwise model, more than 8% over the pipeline counterpart). We list the results of decomposed *Binding* (TRIGGER,TRIGGER) pairs and *Binding* full events of RUPEE, its pipeline counterpart, the pairwise model, the winner in 2013, the best runner up and the system that achieved best result on *Binding* events in Table 4.7. Comparing three models proposed by us, the recalls and precisions of decomposed pairs and full events of *Binding* events are positively correlated. It is because they used the same classifier and features in *Binding* THEME fusion step. RUPEE got the best among these three models because of a better calibration between precision and recall in *Binding* pair extraction. TEES 2.1 and EVEX have the same behavior, but EVEX has worse precision in pair extraction but better precision in full event extraction than RUPEE. It suggests that our weak *Binding* result is cause by *Binding* THEME fusion step. BioSEM, which got the best result on *Binding* event, is a rule based system that extract events through learned patterns in predefined forms. These patterns extract multi-argument events jointly under the restriction of predefined forms. Different to the remarkable high recall in full event extraction, recall on pair extraction of BioSEM is just at average level among the listed approaches. As a rule based system, it achieved the best precision in both pair extraction and full event extraction. We believe that the success of BioSEM is because *Binding* events require to be jointly extracted and their predefined symbolic forms lead to a good balance between precision and recall.

Figure 4.3 displays the precision-recall curves for total (TRIGGER, THEME) pair extraction of all the classes, that is, before the post-processing that is common to the two approaches. We believe that the difference between pipeline counterpart and RUPEE on development set is statistically significant, which is consistent with their performances on test set. The positions and the F_1-scores of the actual classifiers are marked in bold, and

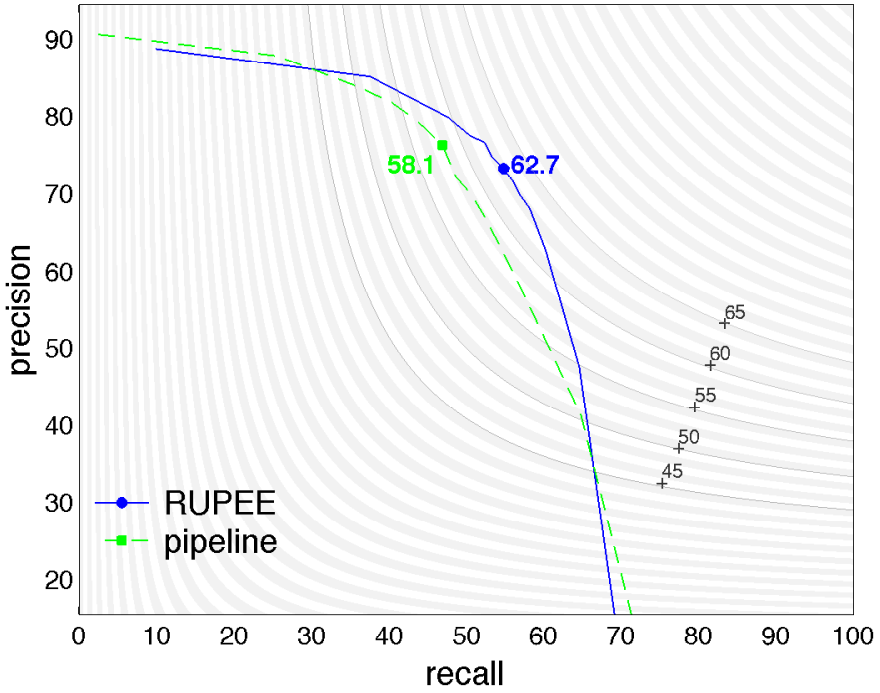


FIGURE 4.3: Precision-recall curves for (TRIGGER, THEME) classification by RUPEE and its pipeline counterpart, on the BioNLP 2013 development set. Level curves of F_1-score are shown in the background.

the level curves of F_1-scores are displayed in the background. Note that these F_1-scores are not necessarily maximal since the classifiers are not calibrated on the test set. The maximal values of recall are moderate, illustrating that present systems fail to retrieve events. Clearly, except for the small values of recall or precision that lead to very low F_1-scores, RUPEE dominates the pipeline model.

4.4.3.3 Training Duration

In this last section, we propose to illustrate the lower complexity of our approach compared to UCLEED by providing durations for training both systems on BioNLP 2011 GE. These timings do not involve preprocessing but only running cross-validation on the training set and evaluation on the development and test sets. For UCLEED, we used the code (in java & scala) provided by the authors on `github.com/riedelcastro/ucleed` and we chose BioNLP 2011 GE because this code was primarily designed to run on it. Our code, in python, is publicly available from `github.com/XiaoLiuAI/RUPEE`. Experiments were conducted on the same computer, with a quad-core Intel Xeon CPU and 16GB of RAM. Both codes are multi-threaded and used all 4 threads simultaneously. Under these conditions, UCLEED requires around *8h30min* to run its 10 epochs, while our code completes training in about *30min*. UCLEED might be faster by using feature caching, but we had to disable it because it was exhausting RAM. Some of these differences may be due to implementation choices, but we believe that the 15-fold speed increase (for around 800 training documents) is at least partially due to the lower complexity of our approach.

4.5 Conclusion

In this chapter, we introduced the RecUrsive Pairwise Event Extraction (RUPEE) model that solves the recursive events extraction problem by dynamically extracting essential pairwise structures from text. This model keeps the principle of pairwise extraction described in the previous chapter, without resorting to the complex inference process of joint models. Thanks to the great improvement on **REG** events brought by recursive extraction, our model reaches the best overall F_1-score reported so far on the BioNLP 2013 and the BioNLP 2011 test sets for a single model. In addition, we merged the pairwise extraction of all the events together, which makes the model simpler than our previous model.

In this model, we simplified the classifier by removing the post-processing of normalized SVM output scores. The experiments shown that this classifier works as well as the SVMs with post-processing. Although it could be possible to improve the performance by using more refined post-processing methods or multi-class classifiers that directly optimizing the global F_1-score, the confusion matrix shows there is little space to improve by this type of tuning.

After solving the recursive events extraction problem, the major drawback of this approach becomes the *Binding* event extraction. Compared to TEES 2.1, EVEX, which solve the multi-argument *Binding* events in similar way as our model, the performance our *Binding* THEME fusion step is relatively poor. Since we used standard binary SVM to solve this problem, using a fine-tuned classifier (SVM with C^+/C^-) is a quick solution to get significant improvements. However, BioSEM, which got the best *Binding* performance, extracts events using patterns in predefined forms that enable to extract *Binding* events jointly. We suppose that better *Binding* event extraction requires a better way to handle the multi-argument (triplet) structure.

Besides development of new model, we also ran experiments to confirm the effectiveness of dependency features. *V-walk*, which is more expressive than *E-walk*, shown surprising effect in pairwise extraction. One possible reason is that *V-walk* features are too sparse because it involves two tokens in triplets. Diverse tokens hurt the generalizability of *V-walk* features, thus *V-walk* features are ineffective in classification. We did not further study the variation of the data matrix of *E/V-walk* features due to the limit of time. The assumption about the sparsity of *V-walk* needs to be confirmed. We suggest to involve continuous vectorial representation of semantic units to solves this problem. In this approach, tokens, dependency tags and even the whole triplet of *E/V-walk* are learned from a large corpus or knowledge base, where similar semantic units have similar vector representations. Besides, it is possible to represent the whole dependency path using semantic composition methods, so that sequence information can be completely kept.

Chapter 5

Variations

5.1 Introduction

In previous chapters, we introduced our best model as well as what are, for us, the optimal features and classifiers for the biomedical Genia event extraction task. After getting this best model and optimal classifiers, we also tested many other solutions, some of which are presented in this chapter. We evaluated most of these solutions directly on the test set. Even if those variants were not necessarily improving the extraction performance, we believe that they carry some scientific interest and could perhaps be eventually successful with more work in slightly different settings. That is the reason why we chose to detail three of them in this chapter.

As described in Section 4.3.1, we did not use the SVMs with post-processing in our best recursive pair-wise model since it does not help to improve the final performance. In Section 5.2, we list the post-processing approaches we tested along with the standard SVMs in one-vs-rest framework. We tested different normalization methods that map the output score into the interval $(0, 1)$, different thresholds and different optimization methods that enable to optimize the F_1-score without normalization. Previous models [20, 65] show that incorporating previous predictions as features for succeeding predictions improves the performance over baseline in sequence labeling. Section 5.3 introduces how we applied this idea by merging the previous predictions into feature vectors following similar principles as previous works.

The sparse data problem is known to be quite pervasive in NLP, because of the atomic symbol representations. The results of our experiments using different dependency path encoding methods indicate that the sparsity problem is also important in the biomedical event extraction task we target. In order to solve it, many methods have been proposed to represent words by continuous vector features, which are learned based on observed co-occurrence patterns. Typical learning approaches try to learn a low dimension representation using techniques such as Clustering, Latent Dirichlet Allocation [66], or Singular Value Decomposition. However, these models are linear, while the semantic relations can be non-linear. Recently, some studies [67, 68] focus on using deep learning techniques, which are non-linear neural networks, to learn these semantic representations. We applied one of the deep learning models to get continuous vector representations from a large unlabeled biomedical corpus and integrated them into the linear classifier of our

recursive pair-wise model. Section 5.4 introduces the details of the deep learning model we used and how we integrated the learned vectors into linear classifiers.

5.2 Classifier

The one-vs-rest framework of SVMs is not consistent, since there is no reason to believe that the scores f_k in (3.3) are calibrated, and thus that the majority vote makes sense. Besides, in this framework, each classifier is trained to optimize a loss function that is different from the F_1-score. As described in Chapter 3, we solved the multi-class classification problem by tuning decision thresholds on the normalized SVM output scores to optimize the micro F_1-score of all the classes. SVMs used in one-vs-rest framework are called *main classifier* in this approach. However, we found that this method does not necessarily bring significant improvement to our recursive pair-wise model. We tested different normalizing methods, thresholds selection strategies and different main classifiers to evaluate their effects, compared to our basic model on the BioNLP event extraction task. We present these experiments in this section.

5.2.1 Output Normalization

In this section, we introduce two approaches that we used to normalize the output scores of the main classifier. Similar to the method used before, both of these two approaches map the output score to the $(0, 1)$ interval through logistic function. The first method was proposed by Duan et al. [58], and is a soft-max method that optimizes the probability of a correct prediction by scaling the output scores of corresponding classes. The other method is a multinomial logistic regression classification, which is similar to a one-vs-rest framework but optimizes a global likelihood instead of a loss function of each class.

5.2.1.1 Soft-Max

Given an example $\mathbf{x}_n$ described by its K-dimensional vector of SVM scores $\mathbf{f}_n$, we can obtain the posterior probabilities through a soft-max function

$$P_k^n = \frac{\exp(w_k \mathbf{f}_n(k) + w'_k)}{z_n},$$

where $\mathbf{f}_n(k)$ is the output score of the kth SVM and $z_n = \sum_{k=1}^{K} \exp(w_k \mathbf{f}_n(k) + w'_k)$ is a normalization term that ensures that $\sum_{k=1}^{K} P_k^n = 1$. The parameters of the soft-max function $(w_1, w'_1), \ldots, (w_K, w'_K)$ can be designed by minimizing a penalized negative log-likelihood function,

$$\arg\min_{\mathbf{w}} \quad E = \tfrac{1}{2}\|\mathbf{w}\|^2 - C \sum_{n=1}^{N} \log P_{y_n}^n \tag{5.1}$$

$$\forall k \in \{1, \ldots, K\}, \qquad w_k, w'_k > 0 \ , \tag{5.2}$$

where $\|\mathbf{w}\|^2 = \sum_{k=1}^{K}(w_k^2 + w_k'^2)$, y_n is the true label of example $\mathbf{x}_n$ and C is a positive regularization parameter. We assume that the score $\mathbf{f}_n(k)$ and the probability of example $\mathbf{x}_n$ is in class k are positive correlated. Therefore, we set positivity constraints on w_k.

By introducing the substitute variables, $s_k = \log(w_k)$ and $s'_k = \log(w'_k)$, the constrained optimization problem can be transformed into an unconstrained one and can be solved using gradient based methods. The first-order derivatives of the objective function with respect to the substitute variables can be computed using the following formulas

$$\frac{\partial E}{\partial s_k} = \frac{\partial E}{\partial w_k}\frac{\partial w_k}{\partial s_k} = \left(w_k + C\sum_{y_n=k}(P_k^n - 1)\mathbf{f}_n(k) + C\sum_{y_n\neq k} P_k^n \mathbf{f}_n(k)\right) w_k \quad (5.3)$$

$$\frac{\partial E}{\partial s'_k} = \frac{\partial E}{\partial w'_k}\frac{\partial w'_k}{\partial s'_k} = \left(w'_k + C\sum_{y_n=k}(P_k^n - 1) + C\sum_{y_n\neq k} P_k^n\right) w'_k \ . \quad (5.4)$$

We used the algorithm L-BFGS-B [69], provided by SciPy[1] to optimize the objective function.

5.2.1.2 Multinomial Logistic Regression

Multinomial logistic regression is a classification method that generalizes logistic regression to multi-class problems. Note that $\mathbf{w}_k$ is the parameter vector for the class k in multinomial logistic regression classifier. Multinomial logistic regression classification method is similar to the soft-max method mentioned above, where posterior probabilities are estimated through a soft-max function except that all the output scores are used for computing each probability

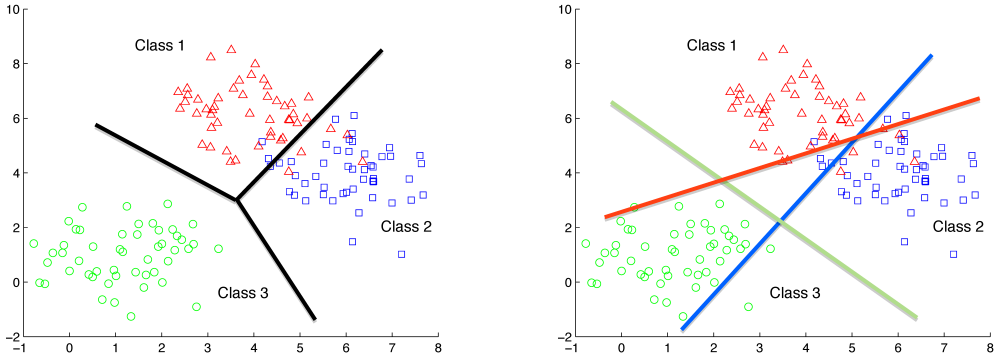
$$P_k^n = \frac{\exp(\mathbf{w}_k \mathbf{f}_n)}{z_n},$$

where $\mathbf{w}_k\mathbf{f}_n$ is the dot product of weighting vector $\mathbf{w}_k$ and the vector of scores $\mathbf{f}_n$, $z_n = \sum_{k=1}^{K}\exp(\mathbf{w}_k\mathbf{f}_n)$ is a normalization term. We used the implementation provided by Statsmodels[2] in our test. This implementation uses a Newton method to optimize the likelihood of the model. The main difference between multinomial logistic regression and logistic regression in a one-vs-rest framework is that multinomial logistic regression imposes the constraint that all the predicted probabilities sum to 1, whereas parameters are estimated in separate models under a one-vs-rest framework. Figure 5.1 shows the difference between multinomial logistic regression and logistic regression in one-vs-rest framework.

5.2.2 Threshold Selection

With normalized output scores, our last step is tuning the threshold to maximize the objective function. Denote g_k the normalized output of class k, we tested two thresholds tuning functions. In paragraph 3.3.1.2, we calibrated the normalized output by adding a threshold to each class, where the final decision is made by $\hat{y} = \arg\max_k(g_k + \theta_k)$. This method calibrates each class with a specified threshold. However, Table 4.4 shows that the vast majority of errors are between *None* class and event classes. Hence, we have a

[1] Available from www.scipy.org.
[2] Available from http://statsmodels.sourceforge.net/



(A) Multinomial Logistic Regression (B) Logistic Regression in One-vs-Rest

FIGURE 5.1: SVM separating hyper-planes

new solution, which only calibrates the difference between *None* class and event classes.

$$g_{\hat{y}'} = \max_{k \neq None} g_k \tag{5.5}$$

$$\hat{y}' = \arg\max_{k \neq None} g_k \tag{5.6}$$

$$\hat{y} = \begin{cases} None & \text{if } \theta \times g_{\hat{y}'} < 1 - \theta \\ \hat{y}' & \text{if } \theta \times g_{\hat{y}'} \geq 1 - \theta \end{cases} \tag{5.7}$$

In this method, we do not change the event class when an event is detected. Moreover, we also tested another method that tunes the transformed output score by a linear function:

$$\hat{y} = \arg\max_k (\theta_k' g_k + \theta_k) \tag{5.8}$$

To optimize the objective function and learn the Θ parameters, we used algorithms that do not need derivatives or second derivatives such as the Brent's method [64] provided by SciPy library. Unfortunately, the tuning function 5.8 always leads to numerical problem during the optimization no matter which optimization method was used.

5.2.3 Without Normalization

In previous sections, we normalized the output scores of SVMs by different logistic transforms to ease the search of decision thresholds. We also tested methods without this normalization, that is, using optimization algorithms that are not sensitive to the interval of output scores of SVMs to optimize the objective function, which is the micro-average F_1-score. All the methods we used are implemented in SciPy.

fmin minimizes the objective function using the downhill simplex algorithm, which was implemented following [70, 71]. Unlike other numeric methods, this method searches the function space by creating simplexes, which is a special polytope of $N + 1$ vertices in N dimension, via heuristics.

fmin_powell minimizes the objective function using a modified Powell's method, which was implemented following [72, 73]. It minimizes the objective function by a bi-directional search along each search vector, which are usually initialized as the normals aligned to each axis. The search result is a linear combination of the search vectors and the displacement of each step is added to the search vectors list. Meanwhile the search vector that is closest to the new search direction is deleted from the search vector list.

fmin_cg minimizes the objective function using a nonlinear conjugate gradient algorithm, which was implemented following the book [74].

fmin_bfgs minimizes the objective function using a BFGS algorithm, which was implemented following the book [74].

5.2.4 Experiments

To observe the variation of the performances of different classifiers, we tested the algorithms with three data-set configurations listed in Table .

TABLE 5.1: Data-set configurations used to test different algorithms.

	Training set	Test set
1)	BioNLP11 train&BioNLP11 development	BioNLP11 test
2)	BioNLP11 train	BioNLP11 development
3)	BioNLP11 train&BioNLP11 development BioNLP13 train&BioNLP13 development	BioNLP13 test

TABLE 5.2: Total F_1-scores for classifiers using different normalizations and thresholds in RUPEE. MNLogit means multinomial logistic regression, LR means logistic regression in One-vs-Rest framework. Thresholds mean the threshold tuning method introduced in 3, thresholds 2 means the method introduced in this chapter.

	BioNLP11 test	BioNLP11 devel	BioNLP13 test
SVMs+LR+thresholds	56.0	53.3	54.2
SVMs+SoftMax+thresholds	-/-	53.5	53.3
SVMs+MNLogit+thresholds	55.7	52.7	53.8
SVMs+MNLogit	55.4	52.3	53.7
SVMs+MNLogit+thresholds 2	55.6	53.1	53.7
SVMs alone	55.6	52.8	**54.4**

Table 5.2 lists the experiments using SVMs with different normalization methods and thresholds. The differences between classifiers using different normalizations on BioNLP11 test set are less than 0.5%, which is not significant. Similarly, the differences on BioNLP11 development set and BioNLP13 test set are also not significant since they are less than 1%. Comparing to the counterpart (SVMs+MNLogit) without tuning decision thresholds, tuning thresholds always improves the performance based on the normalized scores. However, the improvements are always less than 0.5% and the final results do not outperform the pure SVMs without any post-processing since the normalization hurts the performance. Table 5.3 lists the results of classifiers that optimize the thresholds without normalization. All of the four optimizing methods returned F_1-scores, which are very

similar to the SVMs without post-processing. We conclude that different classifiers do not produce significant difference on the final F_1-score for our recursive pair-wise model.

TABLE 5.3: Total F_1-scores for classifiers using different optimization methods in RUPEE.

fmin	fmin_powell	fmin_cg	fmin_bfgs	SVMs alone
54.4	53.8	**54.4**	**54.4**	**54.4**

Since the classifiers with post-processing are reported helpful in previous studies, we ran experiments again with the traditional pipeline model. We substitute the classifier only for the trigger detection step in the pipeline models in these experiments. Unlike the experiments listed above, classifiers with post-processing achieve remarkable improvements of the pipeline model.

TABLE 5.4: Total F_1-scores for different classifiers with Pipeline model

	BioNLP11 test	BioNLP11 devel	BioNLP13 test
SVMs+LR+thresholds	-/-	50.8	52.2
SVMs+MNLogit+thresholds 2	**53.5**	**51.5**	**52.6**
SVMs	50.0	48.7	50.8

Table 5.4 lists the results of pipeline models using different classifiers on different data-sets, where *thresholds* is the method we used in Chapter 3 and *thresholds 2* is the method introduced in this chapter. We did not run all the optional classifiers on all the data-sets, but these results seem to be enough to prove the effectiveness of post-processing for a pipeline model.

In the end, we compared the results of standard SVMs and SVMs with asymmetric costs. The final F_1-score of recursive pair-wise model using standard SVMs on BioNLP11 test set is 49.7 whereas the final F_1-score of the same model using SVMs with asymmetric costs is 54.4. The improvement of using asymmetric costs is very large for our model on this task.

5.3 Stacked Model

Inter-dependencies between recursive events and relations among arguments in multi-argument events may provide useful information for information extraction; it might be hence helpful for a system to know the states (or labels) of adjacent nodes/edges to avoid violating some constraints and improve predictions. As mentioned in 2.4.2.1, the joint system UCLEED represents events as graphs using tokens as nodes and actually uses a global view of the sentence graph to label each of its words. In order to find a global optimum, an iterative dual decomposition method is used to infer the labels of nodes and edges using labels predicted at a previous iteration. Figure 5.2 presents an example, where each circle represents a candidate entity or protein in the sentence and $\{T'_A, T'_B, T'_C, T'_D, T''_A, T''_B, T''_C, T''_D\}$ are labels predicted at each iteration. Suppose node B is a *Binding* trigger and nodes $\{C, D\}$ are proteins. The predicted labels from the first iteration for triggers $\{T'_B, T'_C, T'_D\}$ and relation between two proteins T'_C and T'_D can help

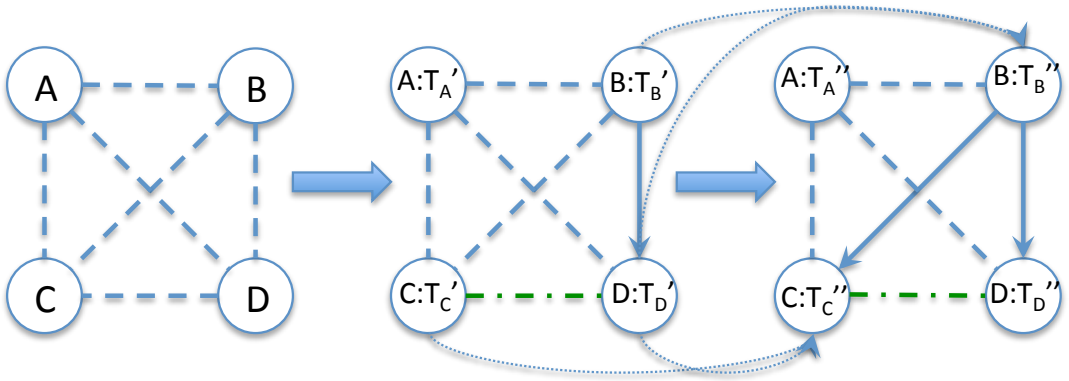


FIGURE 5.2: **UCLEED** inference process. Each iteration makes a full prediction on all the nodes and edges. The arcs represent how the previous predicted labels are incorporated in the next prediction. The green line refers to the prediction of multi-argument *Binding* events.

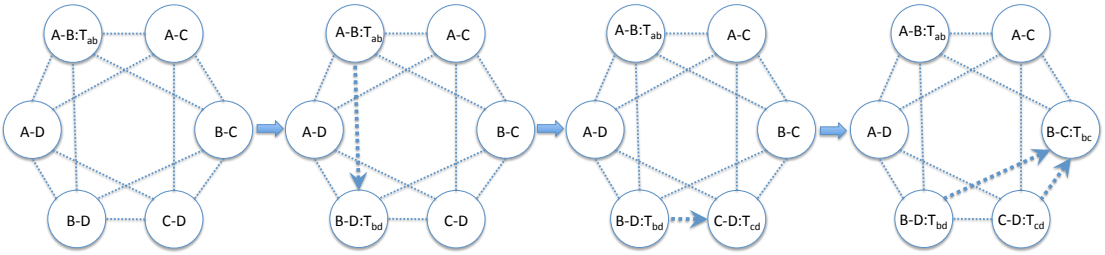


FIGURE 5.3: **SEARN** search process. Each node represents a pair of tokens and edges between nodes represent the potential dependencies between pairs.

the detection of the relation between $\{B, C\}$ in the second iteration. Despite powerful, this approach is computationally expensive.

Another way, called *stacked or greedy learning*, can actually be used to try to improve predictions by using information coming from labels from adjacent nodes and edges. Search-based structured prediction (SEARN) algorithm proposed by [65, 75], is one of this kind of meta-algorithm designed to deal with any structured prediction problem. SEARN first decomposes the structured prediction problem into a set of traditional classification problems, then predicts each problem in turn where the tth decision may be dependent on any of the preceding $t-1$ decisions. Figure 5.3 shows an example of predicting process of SEARN, where a graph prediction problem is decomposed into a set of pairwise relation prediction problems presented in each node. Connections between nodes represent the potential ways how pairwise examples can use the predicted labels for other pairs. In the last graph of Figure 5.3, the pair **B-C** uses the predicted labels of pairs **B-D** and **C-D**. As described above, the inference models also use labels of previous predictions. The major difference between the global inference methods and the stacked learning methods is that the global inference methods use labels predicted in previous iteration, where a complete graph is predicted; stacked learning methods incorporate the predicted labels of decomposed problems, hence do not require the prediction of whole graph. Since our recursive pairwise model applied a dynamic process to extract the (TRIGGER,THEME) pairs, the greedy approach of stacked learning seems like a natural choice that fits nicely in our framework. At test time, to extract events from a fresh

sentence, this strategy is simply applied by running several iterations of our system over the sentence, while using predictions from previous iterations as features for subsequent ones. Such features coming from predictions are simply added to the feature vectors that are fed to the SVMs. We added two kinds of features of previously predicted events:

1. *Neighbor*: an indicator of whether a word appearing in the neighboring tokens of a candidate trigger has been labeled as event trigger at a previous step.
2. *Argument*: an indicator of whether the candidate argument of an event has been marked as event in a previous prediction step.

5.3.1 Simulate the Previous Predictions in Training

With the assumptions of stacked learning, *i.e.* each decision may be dependent on any preceding prediction; the classifier should be trained with any label of preceding decisions that it might encounter. This introduces a chicken-and-egg problem: how to best train a classifier depends on the classifier itself. In this case, simulating the predictions during training step is required since the training data only give examples, where the stacked features are true gold labels, which is different from situations occurring during test. Since our system used SVMs in one-vs-rest framework as classifier, a straight solution is applying learning methods that do not require knowing all the examples at the beginning to train the classifier. On-line learning methods and batch learning methods are often used in this case. The principle of on-line learning is to update a sequence of functions $f_1, f_2, \ldots$, in a way such that the function f_{t+1} depends only on the previous function f_t and next data point (x_{t+1}, y_{t+1}). Suppose that we decompose a structured prediction example (x, y) into a set of traditional classification problems $\{(x, y_i)\}_i$, on-line learning methods enable us to train the classifier f_{t+1} based on the example $([x, (\hat{y}_1, \ldots, \hat{y}_t)], y_{t+1})$, where $\hat{y}_t$ is previous predicted label returned by f_t. Unlike on-line learning methods, the batch learning methods update the functions when given a whole new batch of training examples.

However, we did not implement such kind of learning methods due to the limitation of time. Alternatively, we simulate the predictions by assigning the true labels to the tokens/entities with respect to the order of generating training data. Therefore, the later generated examples incorporate the true labels of early-generated examples into feature vectors, whereas the early-generated examples do not have any information about true labels. Recall that our training set is created following the formula below:

$$\mathcal{P}_S = \{p_{ij} = (t_i, a_j) | t_i \in \mathcal{T}_S,\ a_j \in \mathcal{A}_S\} \cup \{p_{i\alpha} = (t_i, e_{\alpha\beta}) | t_i \in \mathcal{T}_S, \exists\beta : \hat{y}_{\alpha\beta} \neq None\},$$

where $\mathcal{T}_S$ is the candidate entity set and $\mathcal{A}_S$ is the protein set in a sentence. The true labels of all candidate entities are initialized as empty at the beginning. During the generation of feature vectors of candidate pairs, true labels of candidate entities are assigned and are known by all the candidate pairs generated later. With features that describe the neighbors or arguments of candidate entities, the later generated pairs incorporate true labels of earlier generated pairs into feature vectors. Therefore, we get a training set, in which some examples integrate true labels of adjacent nodes, whereas some examples do not. Since that all the structured prediction problems of a single sentence are represented by (x, y), the training set we get are $([x, (y_1, \ldots, y_{i-1})], y_i)_i$ and the order of example sequence is random. This is different to the real test process since we use the true label instead of predictions and the prediction order matters in test.

5.3.2 Evaluating Examples with Different Orders in Test

In our recursive model, for a given sentence, candidate pairs are dynamically added to a candidate set during event prediction; the system samples pairs at random from it to evaluate them and eventually labels them as event. If we start to use features based on previous predictions of the system, then the order with which pairs are chosen from the candidate matters, because different orders might induce different features and eventually different performances.

Therefore, we studied two ways of ordering the choice of examples during test. The first option, the simplest, still consists in choosing the pairs randomly. The second option, more sophisticated, uses the confidence of the classifier to order its predictions. There are two kinds of pairs: pairs between candidate entities and proteins $(t, protein)$, pairs between candidate entities (t_i, t_j). In this second method, we evaluate the $(t, protein)$ pairs randomly since their features are not impacted by previous predictions in our setting, but evaluate the (t_i, t_j) pairs ordered by the score returned by the classifier: we compute the score that would be returned by the SVMs for each of such pairs and only keep the prediction for the one with the top one. This second option is more computationally costly but might intuitively achieve better performance since the system might be able to label "easy" pairs first and then to use these labels to perform predictions on more complex cases. Note that, for both options, we always eliminate candidate pairs to avoid circles in the event graph.

5.3.3 Experiments

We tested two types of stacked features and two evaluation methods on the BioNLP13 test set.

TABLE 5.5: F_1-scores of stacked models with different features and evaluation orders on the BioNLP13 test set

	Argument	*Argument+Neighbor*	Original Model
Random evaluation	53.9	53.9	54.4
Ordered with confidence	54.1	54.0	

Table 5.5 presents the results of models using stacked features, where *Argument* refers to the feature that indicates the event type when a candidate argument was previously predicted as a trigger and *Neighbor* refers to the feature that indicates the event type when a neighboring token belongs to a previously predicted trigger. Unfortunately, none of them improves the performance compared to the original experiment that we reported in previous chapters. This identical performance can be explained using two hypotheses: 1) our simulation method to emulate these features in training is unsuccessful; 2) the stacked features we used are inappropriate for this task. In our recursive pair-wise model, the information of previous prediction is actually implicitly incorporated into the process when the predicted triggers are used as arguments. The label of neighboring tokens seems to be mostly useless since events are rare and scattered in sentences and neighboring labels of a trigger are almost always *None*.

These results confirm why our pairwise system is able to perform identically as the joint model UCLEED: the use of a global optimization approach is not necessarily mandatory to achieve good performance on this task.

5.4 Vector Embedding

Traditional encoding methods represent symbols by sparse vectors, where each value of the feature vectors indicates the frequency, occurrence, *etc.* of a certain symbol indexed in a dictionary. For example, representing a single word requires a feature vector that contains only one non-zero value but has the dimension of the whole dictionary (usually on the the order of 10^5). Indices of atomic symbols in a dictionary do not represent their semantics and relations between them since the product of vectors of two different words is always zero; even if additional features such as POS tags or stem can used to improve this, this is still far away from a semantic encoding. Complex symbol systems that have similar structures are represented as completely independent items, which lead to the sparse data problem.

Learning low-dimensional vector representations (or embeddings) has been proposed as a solution to this issue [66–68, 76–78]. Bordes et al. [76] learn the vector embeddings based on a knowledge base, in which words and symbolic relations and concepts are represented as a big graph. Their method learns words embeddings along with predefined symbolic relations through a neural network. One difficulty of applying this approach into text processing applications is word disambiguation, since one word can refer to multiple semantic nodes in a knowledge base. Another category of methods [66–68, 77, 78] learn the vector embedding from unannotated corpus, mostly based on the occurrence of words. Unlike traditional encoding methods (sparse and large), these embedded vectors are dense and low-dimensional, with a dimension hence independent of the dictionary size. Another important difference between traditional encoding methods and words embeddings is that embeddings are not independent: words with similar semantics have similar embedding vectors. Both reasons make word embeddings an appealing choice to deal with the sparsity problem. With such representations, it is possible for algorithms to learn more complex relations between words without handcrafted heuristics created by experts.

5.4.1 Language Model

One category of methods that learn word vectors is language models. This kind of methods try to learn embeddings from unannotated corpus basically follow the hypothesis that words frequently occurring in similar contexts are semantically similar. Differences (computed by the Euclidean distance for instance) between learned embeddings of semantically similar words is supposed to be smaller than those between vectors of irrelevant pairs of words. Two advantages of language models are: (1) they are unsupervised, (2) embedding vectors naturally solve the sparsity problem without any handcrafted heuristic such as prefix, suffix, stem, POS, *etc.*

Plenty of language models have been proposed in recent years, we selected the model proposed by Huang et al. [68] to learn our words embeddings on biomedical texts. This model considers both local and global contexts to learn the representations of words from

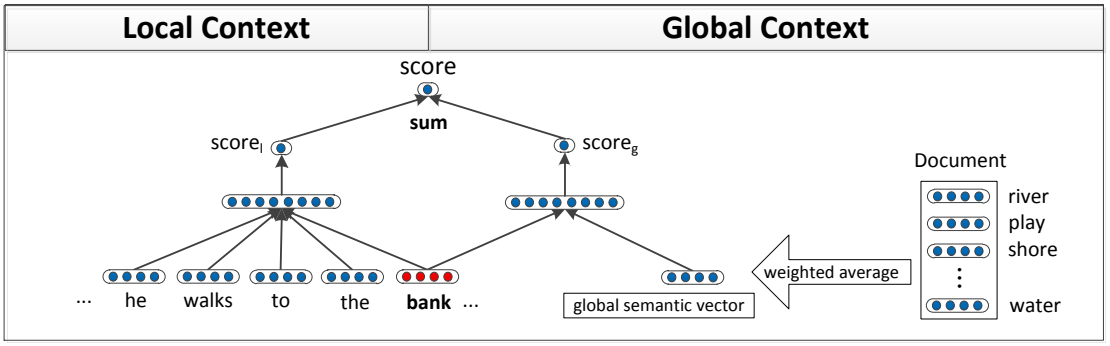


FIGURE 5.4: Language model from [68] (figure extracted from the original paper).

a corpus. Though it can learn multiple embeddings per word to solve the homonymy and polysemy problems, we only used a single prototype per word.

Training Objective and Method In a corpus, sequences of words s with their corresponding documents d are observed, where the size of all sequences is fixed and predefined as the window size. We learn a scoring $g(.,.)$ which is trained to give a value of the adequacy of sequence s and document d. $g(.,.)$ is trained in the following manner. For an observation (s,d), scores $g(s,d)$ and $g(s^w,d)$ are computed, where s^w is the sequence s with the last word replaced by a word w sampled at random in the vocabulary. Following Collobert and Weston [67], $g(s,d)$ is expected to be larger than $g(s^w,d)$ by a margin of 1, because s^w is constructed so that it is a negative example, *i.e.* an invalid English sentence. This corresponds to the training objective of minimizing the ranking loss for each (s,d) found in the corpus:

$$C_{s,d} = \sum_{w \in V} \max(0, 1 - g(s,d) + g(s^w,d)) \ .$$

Training of this objective function is carried out by sequentially randomly choosing pairs (s,d) from the corpus and words from the dictionary to create corrupted examples. Following Huang et al. [68], we used mini-batch L-BFGS [79] with 1000 pairs of valid and corrupted examples per batch for training.

Neural Network Architecture $g(.,.)$ is parameterized by a neural network. Word embedding vectors are stored in an embedding matrix $\mathbf{L} \in \mathbb{R}^{n \times |V|}$, where n denotes the dimension of the vector representations and $|V|$ the size of the vocabulary. With $ind(w)$ returning the index of word w, the ith column of this embedding matrix $\mathbf{L}$ is the vector of word w where $ind(w) = i$. This embedding matrix is learned and updated during training of $g(.,.)$.

Figure 5.4 shows the neural network of the language model of [68] which we used. A word sequence $s = (w_1, \ldots, w_m)$ of length m is represented as the concatenation of the m word embeddings $\mathbf{x}(s) = (\mathbf{L}_{ind(w_1)}, \ldots, \mathbf{L}_{ind(w_m)})$ where $\mathbf{L}_j$ denotes the jth column of the matrix $\mathbf{L}$. The score of local context g_l is computed by a two-layer neural network:

$$\mathbf{a}_l(s) = f(W_1^l \mathbf{x}(s) + b_1^l) \tag{5.9}$$

$$g_l(s) = \mathbf{w}_2^{l\top} \mathbf{a}_l(s) + b_2^l \ . \tag{5.10}$$

where $\mathbf{a}_l \in \mathbb{R}^h$ is the activation of the hidden layer with h hidden nodes, f is an activation function such as (tanh in our experiments), $W_1^l \in \mathbb{R}^{h\times(mn)}$, $\mathbf{w}_2^l \in \mathbb{R}^h$ are weights of the first and second layers of the neural network and b_1^l, b_2^l are the biases of each layer.

For the global score g_g, the document $d = \{w_1, \ldots, w_k\}$ is represented as a weighted average of its word embeddings:

$$\mathbf{c}(d) = \frac{\sum_{i=1}^{k} h(w_i)\mathbf{L}_i}{\sum_{i=1}^{k} h(w_i)} \ ,$$

where $\mathbf{L}_i$ is the embedding vector of the word w_i, $h(w_i)$ is the weight of the word w_i. In our case, idf-weighting is used as the weighting function. Then, the global context score g_g is computed by a two-layer neural network:

$$\mathbf{a}_g(s,d) = f(W_1^g[\mathbf{c}(d); \mathbf{x}(s)_m] + b_1^g) \quad (5.11)$$

$$g_g(s,d) = \mathbf{w}_2^{g\top}\mathbf{a}_g(s,d) + b_2^g \ . \quad (5.12)$$

where $[\mathbf{c}(d); \mathbf{x}(s)_m]$ is the concatenation of the weighted average document vector $\mathbf{c}(d)$ and the vector of the last word of s. All the other parameters in Equation (5.11) and (5.12) are similar to the Equation (5.9) and (5.10).

The final score is the sum of the two scores:

$$g(s,d) = g_l(s) + g_g(s,d) \ .$$

The local score preserves word order and syntactic information, while the global score uses a weighted average, which is similar to bag-of-words features, capturing semantics and topics of the document.

5.4.2 Integration into NLP Tasks

There are two ways of integrating word vector embeddings into a NLP system, either by replacing traditional binary word representations by word embeddings, or by creating supplementary high level features based on the embeddings. The first method inserts the word embedding vectors directly into the features, which are typically useful for low-level NLP tasks like tagging and NER. Models designed for these tasks usually focus on the local context and the morphological features of words, which could be replaced by the word embeddings. Similarities between word embeddings (semantically similar words have similar embeddings) could improve generalization capabilities of such models and perhaps allow the application of nonlinear models. However, this simple method is limited for high-level NLP tasks like syntactic parsing, information extraction, which require to take structured outputs into account and to consider global contexts. For example, for event extraction, the most important dependency path features can not be well represented by word embeddings directly. Since the dependency path features are crucial in event extraction tasks, simply combining the traditional representation of dependency path features and word embeddings might not be beneficial. Previous work [80–83] proposed to compose the semantic representation of a whole sentence by combining word embeddings. This kind of methods shows the possibility of building high-level features using word embeddings along with structured information.

Due to the limited duration of this thesis preparation, we unfortunately did not study the methods that could represent the dependency path using a combination of embeddings. Alternatively, we used the linear classifier trained with our heuristic features along with word embeddings, where the stemmed token features for head tokens and neighbor token features of candidate trigger entities and argument entities were replaced by embeddings. All other features were kept identical as for experiments of previous chapters.

Using directly the low-dimensional vector embeddings in the features might not be directly beneficial since the similarity information they contain might be hard to exploit by a linear classifier, especially when this low-dimensional dense features are combined with the other features (dependency path, *etc.*), which are high dimensional and sparse. Hence, we also tried to quantify these features using a K-means clustering of the word embeddings. In this approach, we aggregated the word embeddings into K cluster and represented each word by a sparse vector, where kth element of the vector indicates whether this word belongs to cluster k or not.

5.4.3 Experiments

In order to train the language model, we downloaded open access articles from *PubMed* matching the keyword *NFkB*, which is the same keyword used to create the BioNLP Genia Task data. We downloaded 2016 documents, which we tokenized using the Stanford tokenizer. We kept a vocabulary of the top $20k$ words with respect to their number of occurrence. The rare words were replaced by **_TOO_RARE_**. To learn the word embeddings, we used the same hyper-parameters as in [68], where the window size is 10, numbers of hidden units in the neural networks for g_l and g_g are both 100 and the dimension n of the word embeddings is 50. As mentioned in Section 5.4.2, we integrate the dense representation into the feature vectors in two ways: the first one replaces the token features by the embedding vectors directly with a scaling factor, while the other replaces the words by sparse vectors generated through K-means clustering on embedding vectors. Since the embedding vectors contain continuous numbers, which differ from the binary sparse representation of the other features, we tried different value of a parameter λ used to scale the features of the embedding vectors: $\mathbf{v} = [\mathbf{v}_{sparse}; \lambda\mathbf{v}_{embeddings}]$.

TABLE 5.6: Total F_1-scores on BioNLP13 test set with embedding features.

Vector Embedding			KMeans		original
scale 0.01	scale 1	scale 100	200 clusters	300 clusters	
54.0	53.5	50.5	53.1	53.9	**54.4**

Table 5.6 displays the results of experiments with different ways of using embedding features. All the models with new features obtain worse performance than the one using old features. For the three experiments using directly the embedding vectors with different scales, the model with the smallest scale gets the closest performance to the original one, which seems to indicate that the embedding vectors are just hurting the performance. For the two experiments using sparse representation derived from clustering, experiment with more clusters performs slightly better.

5.4.4 Perplexity of Language Model with Respect to Annotation

Since the experimental results shown that embedding vectors hurt the performance, we measured the perplexity of the language model with respect to the gold annotation of training set and development set from BioNLP11 and BioNLP13.

As mentioned above, the main hypothesis behind a language model is that words occurring in similar contexts have similar meanings, which is a standard hypothesis in distributional semantics. However, it is clear that this hypothesis can be far from the true except the homonymous and polysemous words. For example, antonyms are very likely to occur in identical environments such as "I like it" versus "I hate it". A purely unsupervised language model cannot capture the difference between "like" and "hate", which should be far from each other in the embedding space. Unfortunately, for our task of event extraction, the polarity of words is crucial for the distinction between *Positive_ regulation* and *Negative_ regulation* events. Another problem is the recognition of the same base form: for example, "activate" and "activation" can have very different vector representations than expected because their syntactic contexts (verb versus noun) are usually quite different. But for biomedical event extraction, one would hope these words have similar representations. Besides, relations between two words are more complex than just similarities. Relations like hypernymy, meronymy are difficult to represent by language models, but these relations can be important to recognize new patterns. Finally, vector representations learned through a unsupervised model are not necessarily correlated to a specific classification task.

Since we use the word embeddings as alternatives for token features, it might interesting to analyze the perplexity of the language model with respect to the event classes. Unlike dependency path features, token features (for trigger words and their local contexts) are very important for classifiers to recognize the trigger event type. In addition, we use linear classifiers, which require learned vector representation to be correlated with the event classes. We evaluated the quality of a learned representation by comparing the similarity of word vectors with their occurrences in biomedical events as given by gold annotations of the BioNLP Genia Task data sets. To simplify the problem, we just observed the word set at the intersection of the trigger words of the gold annotation $\mathcal{V}_{gold}$ and the $20k$ words learned in language model $\mathcal{V}_{lm}$. The intersection is denoted $\mathcal{V} = \mathcal{V}_{gold} \cap \mathcal{V}_{lm}$ and contains $|\mathcal{V}| = 1033$ words.

For each word $w_i \in \mathcal{V}$, we compute a correlation coefficient between two ranked lists l_i^{lm} and l_i^{gold} that contains the most similar words with respect to the language model and events from gold annotation respectively. The ranked list for the language model l_i^{lm} is generated by sorting all the words $w_{j \neq i} \in \mathcal{V}$ based on their Euclidean distance with the word of interest w_i, given by $||L_{ind(w_i)} - L_{ind(w_j)}||_2$. We only keep p most similar words in the list. The ranked list for gold annotation l_i^{gold} is generated by re-sorting the p words retrieved in l_i^{lm} based on the *class-dependent similarity*, which is the sum of products between the frequencies of two words appearing in each class $sim_{ij} = \sum_k (\frac{n_{ik}}{\sum_k n_{ik}} \frac{n_{jk}}{\sum_k n_{jk}})$ where n_{ik} is the number of appearance of word i in class k. With those two ranked lists, we can judge the correlation between the language model and event annotation for words from $\mathcal{V}$ using existing ranking correlation metric.

We selected Spearman's rank correlation coefficient to measure the correlation of the two ranked lists. The Spearman correlation coefficient is defined as the Pearson correlation coefficient between the ranked variables. If there are no repeated data values, a perfect

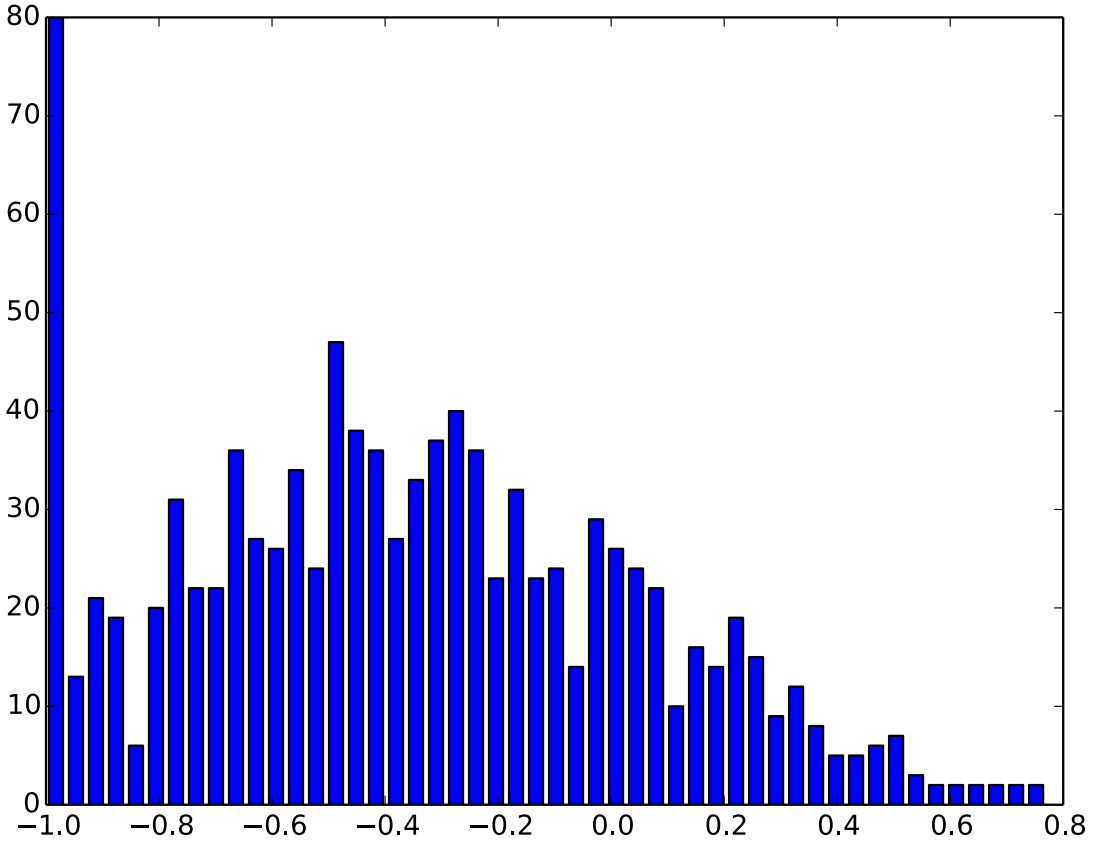


FIGURE 5.5: Histogram of trigger words with respect to Spearman correlation

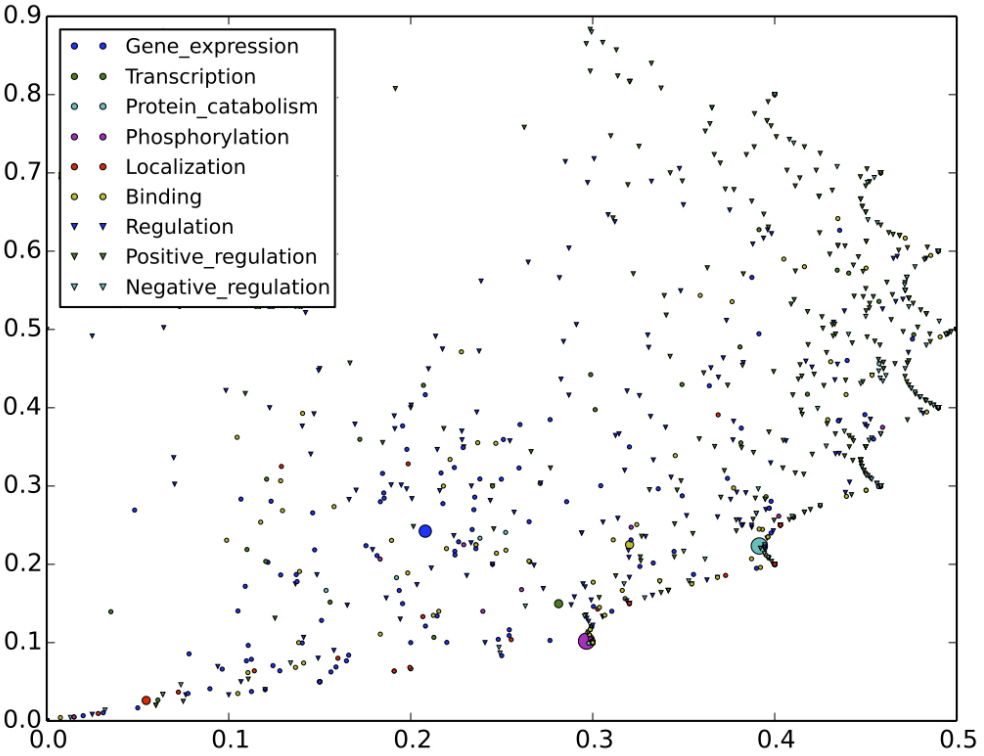
Spearman correlation of $+1$ or -1 occurs when each of the variables is a perfect monotone function of the other. For a sample of size n, and $\mathbf{A}$ and $\mathbf{B}$ are two ranking lists of n objects, this coefficient is computed by:

$$\rho = 1 - \frac{6 \sum_i d_i^2}{n(n^2 - 1)},$$

where $d_i = \mathbf{A}_i - \mathbf{B}_i$ is the difference between two lists at position i. Figure 5.5 presents the histogram of Spearman's rank correlation coefficient for the 1033 observed words. The correlation coefficients of most of the words are negative, which means similar words learned by the language model rarely belong to the same event class defined by the gold annotation. This might explain why using the language model does not bring any improvement to our event extraction system.

Sometimes, the rank correlation coefficient can not correctly measure the perplexity. Indeed, if class-dependent similarities between one word and all its p nearest words given by language model are very high, the relative order of these nearest words should be ignored. Hence, we conducted another analysis by treating the class-dependent similarities of nearest neighboring words as sets of numbers instead of ordered lists. Given a word w_i and p nearest words $(w_{i1}, \ldots, w_{ip})$, we have a vector of class-dependent similarities between them $\mathbf{sim}_i = (sim_{i1}, \ldots, sim_{ip})$. Based on these p similarities, each word w_i can be represented by two scaled values $m_i = \overline{\mathbf{sim}_i}$, the mean of this vector, and $\sigma_i = \sqrt{\frac{1}{p-1} \sum_{j=1}^{p} (sim_{ij} - \overline{\mathbf{sim}_i})^2}$, its standard deviation. High values of m_i indicate that most of the nearest words participate in nearly the same event as the observed word.

FIGURE 5.6: Distribution based on nearest words, size of marker refers to the proportion in the class

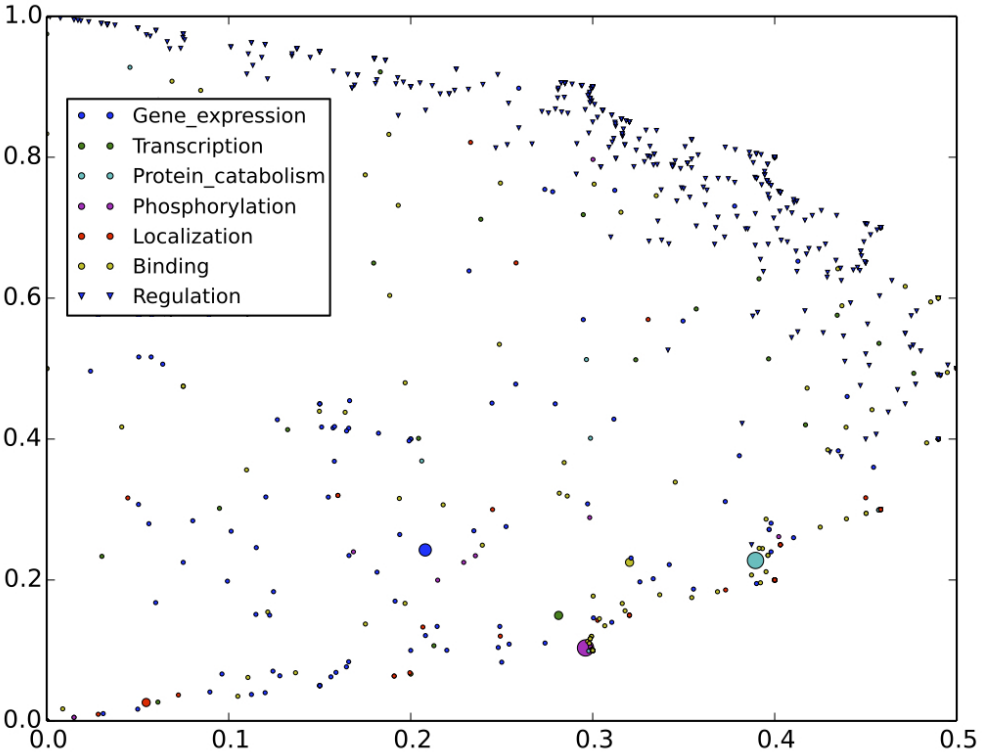


The standard deviation measures the stabilization of the class-dependent similarities of the nearest words.

Figure 5.6 represents the distribution of words in a space where the X-axis depicts the standard deviation σ and Y-axis the mean m. Points in upper-left corner have the best behavior since they have high average similarity and low variance. On the contrary, the bottom-right points corresponds to badly represented words because their nearest words mainly occur in different classes. In order to well present the distribution of words for each class, we indicate the *dominant class* of each word by plotting them in different colors and styles. Class k is the dominant class of a word when this word is most frequently assigned to this class. Besides, the importance of word w_i to class k is scaled by its proportion in this class $\frac{n_{ik}}{\sum_j n_{jk}}$ where n_{jk} is the number of word w_j occurs in class k. The more important a word is to its dominant class, the bigger is its symbol drawn. The sizes of markers shown that the distribution of trigger words of **REG** events are really scattered, whereas other event types have some key words that play a very important role. However, those important words are badly represented in the language model since the big markers of *Gene_expression*, *Phosphorylation*, *etc.* are all in the bottom of figure, which means that their nearest words frequently have other dominant event types.

As introduced before, the language model cannot capture the polarity of words, so it can be more appropriate to judge the quality of word representation while treating the three **REG** events together. Figure 5.7 shows the new distribution of words, for which we merged the three**REG** events into one event. This time, the regulation trigger words are

FIGURE 5.7: Distribution based on nearest words, size of marker refers to the proportion in the class, regulation events are merged into one event



much better represented, which proves that the regulation trigger words mainly appear in similar context.

Overall, this analysis suggests that integrating word embeddings directly into a linear classifier cannot improve the performance. The experimental results with word embeddings are worse than those without them, which is, unfortunately, consistent with our analysis.

5.5 Conclusion

This chapter introduced optional solutions we tested besides our best solution. The post-processing of classifier tries to optimize the F_1-score instead of the original classification objective function. Since the pipeline model benefits a lot from this post-processing, we can say that, in this case, the training objective (single-class classification) is much different from the task desired F_1-score and that a global post-processing is needed there. In contrary, the post-processing does not improve the performance of our recursive pairwise model, which indicates that the classification problem of our model is very close from the final problem. Solving the multi-class classification problem defined by our model seems to somewhat directly optimize the F_1-score.

We also tried to use the previous predictions as features for successive predictions. But the experimental results show that our solution does not work properly. It may be caused by two reasons: we replaced the on-line learning by an ineffective simulation and we encode labels of previous predictions into feature vectors for linear classifier. As

UCLEED and SEARN systems both reported that involving the information of previous predictions can improve the performance, it is a direction that worth further study, but maybe within a global input-output learning framework.

In order to solve the feature sparsity problem, we used a language model to learn continuous vector representations of words from an unlabeled corpus of biomedical text. However, the learned vector embeddings are not well correlated with the final task. If one wants the learned vector embeddings to play the same role as the traditional representations, the underlying similarity hypothesis of language model is ineffective for biomedical event extraction since the task requires very precise distinction between words. For example, the distinction between different kinds of triggers relies on the deep semantics of words more than on their syntactic roles. Besides, words that share the same base form usually have different local contexts in documents, which causes the vector embeddings learned by language models not to be similar enough. We suggest to integrate the existing knowledge base into the representation learning, because the words and relations in knowledge base are richer. Unlike for general text, domain specific terminologies are usually unambiguous in context. Representing words by base form as well as POS is enough as the verb and noun of the same base word always indicate the same thing.

Apart from the quality of word representations, the method of employing them into event extraction task is also important. Continuous vector representations are learned non-linearly and feeding them to a linear classifier may be sub-optimal. Besides, the major drawback of the encoding methods for event extraction task is more the representations of dependency paths than those of trigger words. Many methods [80–83] have been proposed to compose vector representations of sentences or paths in syntactic trees. Learning a dense representation of the dependency path or other structured information based on the word embeddings could be more helpful than using the word embedding directly for event extraction tasks.

Chapter 6

Conclusion

6.1 Perspectives for Biomedical Event Extraction

6.1.1 Contributions and Limitations

Throughout this thesis, we have exhibited a robust model to handle the BioNLP Genia event extraction tasks. As a trade-off between pipeline and joint models, our model reaches a good balance between extraction performance and computational complexity. Our model achieves the best performance on the BioNLP2013 test set and slightly outperforms the best joint model UCLEED on the BioNLP2011 test set. Besides, our model is much faster to train than UCLEED (30 minutes versus 8 hours 30 minutes).

Our model extracts pairwise relations, which is a relevant structure for most events, but not for the multi-argument events, such as *Binding*. As most traditional NLP approaches, we use linear classifiers on heuristically derived features that are based on the morphology and syntax of language. A major drawback of these features, based on symbolic representations, is that they generate sparse vectors that lead to difficulties for learning tasks related to structured information extraction. Besides, we did not address the coreference problem, which should be helpful for recognizing the relationships between entities that explicitly occur far away from each other.

6.1.2 Further Extensions

Our model can be improved in two directions: 1) generalizing the model to multi-argument events and coreferences, 2) creating new features. Our model treats the multi-argument events, which are triplets, as three pairwise relations. Similar to the n-grams representation of sequence, representing triplets by pairs loses some of the joint information. BioSEM, a rule-based system that learns patterns that encode triplets directly, achieves the best *Binding* event extraction performance so far. Representing the triplet directly in dependency parse tree/graph is a promising way to catch the ternary relations. Besides, Yoshikawa et al. [84] use the transitivity of coreference relationship to improve the performance of biomedical event extraction. They substitute the arguments by the nearest coreferences in the extraction of pair-wise relations between triggers and

arguments. Their results shown that using coreference can significantly improve the performance of biomedical event extraction.

The other direction is using embedding vectors learned by language models to represent the dependency path. With the word embedding vectors, there are many potential ways of representing the semantics of a sentence through dependency path. Under the assumption that a sentence can be represented in the continuous vector space of word embeddings, many methods were proposed to construct a representation of sentences based on the representation of words. Dinu et al. [85] reviewed the methods proposed before, where the composition of words by addition, multiplication, *etc.*, may be used to represent a part of a dependency parse tree. Another kind of methods would take both the words and dependency relations into account, which can be seen as somewhat similar to the representation of pairwise relations in knowledge base proposed by Bordes et al. [76].

6.2 NLP Directions

In the past decades, natural language processing has been greatly developed and widely used in many commercial applications. However, for tasks that require the deep understanding of a complex context, the performances of current NLP systems are far from humans.

The main stream of current NLP approaches combines machine learning with handcrafted shallow features. Segmentation and tagging tasks are essentially solved since current models have reached near-human performance (more than 90%). The same principle is followed to address information extraction tasks by considering information related to larger contexts through heuristic shallow features, which encode the syntactic information from parsing results. Unfortunately, these methods do not reach near-human performances. It is because statistical and rule-based methods only focus on the language morphology and language syntax but not on language meaning and language context.

For the well-solved tasks, a large proportion of examples can be processed by simple language morphology and language syntax, whereas most of problems in information extraction tasks require to understand language meaning and language context. For example, most cases in the named entity recognition tasks can be recognized by the words morphology and limited keywords, whereas a small part of exceptions need the comprehension of language meaning. These exceptions may be processed by case enumeration (dictionary) since their are few. This approach does not scale to information extraction problems. In fact, through the discussion with other NLP researchers, many researchers believe that good dictionaries created manually or automatically from high-quality resources are required for NLP applications. I agree with this conclusion because the meaning of words is unpredictable in general, which means looking new words up in the dictionary is necessary to correctly interpret the natural language. Even though some rules can be derived from the morphological feature to infer the grammatical properties for rigorous languages, such as Germany or French, the evolution of natural language and cross-culture communication continuously introduce exceptions. For languages that do not split the words by white spaces like Chinese, a good dictionary is crucial for models to make correct segmentations for even common words, where the impact of out-of-vocabulary words is five times [86] higher than the ambiguous words for the performance

of segmentation. Case study is also very important for machine translation during word alignments. For information extraction tasks, case enumeration is inefficient due to the cheer number of possibilities. Traditional methods use the syntactic structure of sentence to extract the desired structured information without directly understanding the semantic. But the syntactic structures are usually used with words together, which are even more complex than word sequences. Grasping representative features from syntactic structures cannot be effectively solved by handcrafted shallow features. Moreover, language semantic and language context are essential for predictions because the ground truth context and background knowledge. Therefore, I believe that the future of information extraction or even the whole NLP domain is in modeling the language semantic and language context.

Unlike the natural signals such as images, sounds or other senses, human languages are symbolic systems created by humans to describe the world. For example, given an image of apple, everyone who ever seen an apple will understand the object; but people do not understand the word when they do not understand the corresponding language. Since natural language is not "natural" but "man-made", it is rational to mimic humans. Considering that humans do not explicitly think about the grammars while reading and speaking, modeling the sentence without any syntactic parsing can be a natural way to simulate how humans process the language. Kalchbrenner and Blunsom [87] implemented machine translation by recurrent neural networks without any syntactic parse tree, showing that it is possible to throw syntactic parsing away in NLP applications. However, incorporating the syntactic parse tree with the semantic information should improve prediction for tasks with small training sets.

6.2.1 Representation

Suppose things are stored as concepts in the brain, humans decode the symbol of language and associate it to the concept stored in their brain when they read a word. Besides, people can represent what they read by completely different sentences besides using synonyms or changing the order of words. That means a complex context (phrase) is compressed into a central idea and decompressed into different concrete representations. It is helpful if machine catches the language meaning by robust representations, which can reflect the concept of the symbol naturally and enable the compression of larger contexts into comparable representations (words can be summarized by explanations).

Continuous vector space representation, where each word is represented by a fixed-size dense vector, and similar words have similar vector representations. Many deep learning methods [67, 68, 77] have been proposed to learn the representation of words from unlabeled corpus through neural networks. These approaches basically follow the principle that words appearing in similar contexts have similar significations. Some works [67, 68] observe the context by word window, while Mikolov et al. [77] uses all the previous words to estimate the next word by recurrent neural network. Note that the word representation learning methods listed above only observe the words, which is not appropriate to deal with phrases such as "hot dog" or "take off". We used the method proposed by Huang et al. [68] to learn the word representations and integrated them into our linear feature vectors. Though our experimental results of using word representations are not good, we still think deep learning is a very potential direction because the dense vector representations of words bring us the chance to construct new representation of the relational features. Following the idea that a sentence can be compressed into a central

concept, works presented in [78, 81–83] illustrate how to use recursive neural networks to compress a sentence into the continuous vector space of words. Though this model may be simplistic for very complex sentences, it allows to represent words and their explanations in a single dictionary. It is possible to use similar methods to represent a sub-sentence or syntactic parse tree and apply this dense representation in the structured information extraction.

6.2.2 Background Knowledge

Domain specific knowledge is important for people to understand professional articles. Logical inference with knowledge base can help the NLP systems to eliminate false interpretations. However, the symbolic representations of knowledge base are difficult to transfer to NLP applications. Deep learning gives us the chance to model the knowledge base in a numerical form that can be used directly by statistical models. However, the word representation learning methods mentioned above are based on very simple assumptions, which forbid to model complex relations between words such as polarities. Our study on the learned word representation demonstrates that representations learned from unlabeled corpus are not precise enough for the biomedical event extraction tasks. I believe that learning representation from the knowledge base could return better representation than from unannotated corpus.

Bordes et al. [76] learn the representations of knowledge base, where concepts and relations in the knowledge base are represented by dense vectors. There is still a gap between NLP applications and learned vector representations of knowledge base: associating words to concepts in the knowledge base requires disambiguation since one word often points to multiple concepts.

Appendix A

Linguistic Knowledge

A.1 Part-of-speech

In grammar, a part of speech is a linguistics category of words, which is generally defined by the syntactic or morphological behavior of the lexicon item.

Table A.1: Part-of-speech tags used in Penn Treebank.

Tag	Description	PRP$	Possessive pronoun
CC	Coordinating conjunction	Tag	Description
CD	Cardinal number	RB	Adverb
DT	Determiner	RBR	Adverb, comparative
EX	Existential there	RBS	Adverb, superlative
FW	Foreign word	RP	Particle
IN	Preposition or subordinating conjunction	SYM	Symbol
JJ	Adjective	TO	*to*
JJR	Adjective, comparative	UH	Interjection
JJS	Adjective, superlative	VB	Verb, base form
LS	List item marker	VBD	Verb, past tense
MD	Modal	VBG	Verb, gerund or present participle
NN	Noun, singular or mass	VBN	Verb, past participate
NNS	Noun, plural	VBP	Verb, non-3rd person singular present
NNP	Proper noun, singular	VBZ	Verb, 3rd person singular present
NNPS	Proper noun, plural	WDT	Wh-determiner
PDT	Predeterminer	WP	Wh-pronoun
POS	Possessive ending	WP$	Possessive wh-pronoun
PRP	Personal pronoun	WRB	Wh-adverb

Almost all languages have the lexical categories noun and verb, but beyond these there are significant variations in different languages. For example, Japanese has as many as three classes of adjectives where English has one; Chinese, Korean and Japanese have nominal classifiers whereas European languages do not; many languages do not have a distinction between adjectives and adverbs, adjectives and verbs or adjectives and nouns, *etc.* This variation in the number of categories and their identifying properties entails that analysis be done for each individual language. Nevertheless the labels for each category are assigned on the basis of universal criteria. Table A.1 lists the common POS tags used in English language. Practical applications may use a subset of these tags.

A.2 CoNLL Dependency Grammar

CoNLL format, which is proposed by the Conference of Natural Language Learning 2006, is widely used in many syntactic parsers. Given a sentence:

Bell, a company which is based in LA, makes and distributes computer products.

An example of incomplete CoNLL representation of the parse tree of this sentence is:

ID	FORM	LEMMA	CPOSTAG	POSTAG	FEATS	HEAD	DEPREL	PHEAD	PDEPREL
1	Bell	-	NNP	NNP	-	11	nsubj	-	-
2	,	-	,	,	-	1	punct	-	-
3	a	-	DT	DT	-	4	det	-	-
4	company	-	NN	NN	-	7	nsubjpass	-	-
5	which	-	WDT	WDT	-	0	erased	-	-
6	is	-	VBZ	VBZ	-	7	auxpass	-	-
7	based	-	VBN	VBN	-	4	rcmod	-	-
8	in	-	IN	IN	-	0	erased	-	-
9	LA	-	NNP	NNP	-	7	prep_in	-	-
10	,	-	,	,	-	1	punct	-	-
11	makes	-	VBZ	VBZ	-	0	root	-	-
12	and	-	CC	CC	-	0	erased	-	-
13	distributes	-	VBZ	VBZ	-	11	conj_and	-	-
14	computer	-	NN	NN	-	15	nn	-	-
15	products	-	NNS	NNS	-	11	dobj	-	-
16	.	-	.	.	-	11	punct	-	-

TABLE A.2: Attributes used in CoNLL Format.

Field number	Field name	Description
1	ID	Token counter, starting at 1 for each new sentence
2	FORM	Word form or punctuation symbol
3	LEMMA	Lemma or stem (dependency on particular data set) of word form, or an underscore if not available
4	CPOSTAG	Coarse-grained part-of-speech tag, where tag-set depends on the language.
5	POSTAG	Fine-grained part-of-speech tag, where the tag-set depends on the language, or identical to the coarse-grained part-of-speech tag if not available.
6	FEATS	Unordered set of syntactic and/or morphological features (depending on the particular language), separated by a vertical bar (|), or an underscore if not available.
7	HEAD	Head of the current token, which is either a value of ID or zero ('0'). Note that depending on the original treebank annotation, there may be multiple tokens with an ID of zero.
8	DEPREL	Dependency relation to the HEAD. The set of dependency relations depends on the particular language. Note that depending on the original treebank annotation, the dependency relation may be meaningful or simply 'ROOT'.
9	PHEAD	Projective head of current token, which is either a value of ID or zero ('0'), or an underscore if not available. Note that depending on the original treebank annotation, there may be multiple tokens an with ID of zero. The dependency structure resulting from the PHEAD column is guaranteed to be projective (but is not available for all languages), whereas the structures resulting from the HEAD column will be non-projective for some sentences of some languages (but is always available).
10	PDEPREL	Dependency relation to the PHEAD, or an underscore if not available. The set of dependency relations depends on the particular language. Note that depending on the original treebank annotation, the dependency relation may be meaningful or simply 'ROOT'.

In this example, some attributes are always empty because the parser does not support relative parsing functions. Table A.2 lists all the attributes defined in CoNLL format. But most of syntactic parsers only support a subset of these attributes. One can construct

a dependency parse tree through the DEPREL and PHEAD tags, which describe the dependency relation type and the head token of current token.

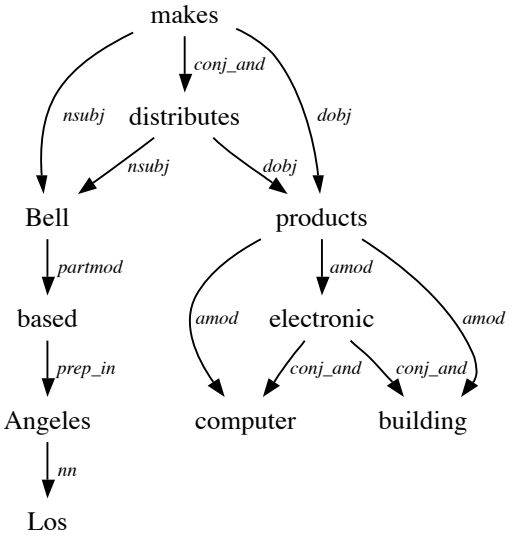
A.3 Stanford Dependency Grammar

Stanford dependency format represent the dependency graph by binary relations between tokens. Here is an example sentence:

Bell, based in Los Angeles, makes and distributes electronic, computer and building products.

For this sentence, the collapsed Stanford Dependencies (SD) representation is:

nsubj(makes-8, Bell-1)
nsubj(distributes-8, Bell-1)
vmod(Bell-1, based-3)
nn(Angeles-6, Los-5)
prep_in(based-3, Angeles-6)
root(ROOT-0, makes-8)
conj_and(makes-8, distributes-10)
amod(products-6, electronic-11)
conj_and(electronic-11, computer-13)
amod(products-16, building-15)
dobj(makes-8, products-16)
dobj(distributes-10, products-16)



where "prep_and" merges the dependency type "prep" and the word "and" to make the parsing result more compact. These dependencies map the sentence straightforwardly onto a directed graph representation, in which words in the sentence are nodes in the graph and grammatical relations are edge labels. Figure in the right gives the graph representation for the example sentence above. The dependency types are listed in Table A.3.

Table A.3: Stanford Dependency Type.

acomp	adjectival complement	advcl	adverbial clause modifier
advmod	adverbial modifier	agent	agent
amod	adjectival modifier	appos	appositional modifier
aux	auxiliary	auxpass	passive auxiliary
cc	coordination	ccomp	clausal complement
conj	conjunct	cop	copula
csubj	clausal subject	csubjpass	clausal passive subject
dep	dependent	det	determiner
dobj	direct object	discourse	discourse element
expl	expletive	goeswith	goes with
iobj	indirect object	mark	marker
mwe	multi-word expression	neg	negation modifier
nn	noun compound modifier	npadvmod	noun phrase as adverbial modifier
nsubj	nominal subject	nsubjpass	passive nominal subject
parataxis	parataxis	pcomp	prepositional complement
pobj	object of a preposition	poss	possession modifier
preconj	preconjunct	possessive	possessive modifier
predet	predeterminer	prep	prepositional modifier
prt	phrase verb particle	prepc	prepositional clausal modifier
punct	punctuation	quantmod	quantifier phrase modifier
ref	referent	root	root
vmod	reduced non-finite verbal modifier	tmod	temporal modifier
xcomp	open clausal complement	xsubj	controlling subject

Appendix B

BioNLP Genia Task

B.1 Task Definition

The BioNLP Genia task requires to extract bio-molecular events mentioned in text, given the protein/gene names, where proteins and genes are not distinguished. It was divided into three sub-tasks:

1. **Core event extraction** task requires to identify the event triggers, their types and the participants of events.
2. **Event enrichment** task requires to extract additional arguments to enrich the core events, such as the location of events. One has to recognize entities of the additional arguments and detect the relations between these entities and the core events.
3. **Negation and speculation recognition** task requires to find negations and speculations regarding events extracted by task 1.

Table B.1 lists the event definitions used in BioNLP 2009 and 2011 Genia tasks. To be consistent with the nomenclature of this thesis, we changed the names of argument groups. We used *participants* instead of *Primary Arguments* in [1, 2] and used *Additional Arguments* instead of *Secondary Argument*. The **core event extraction** task requires to extract the events with their participants while the **event enrichment** task requires to extract the additional arguments. The **negation and speculation recognition**

TABLE B.1: BioNLP Genia event extraction task definitions. "?" means that this argument is optional, "+" means the this argument can occur more than once.

Event Type	Type of Theme participant, other participants (type)	Additional Arguments
Gene_ expression	Protein	
Transcription	Protein	
Protein_ catabolism	Protein	
Phosphorylation	Protein	Site (Entity)?
Localization	Protein	AtLoc (Entity)?, ToLoc (Entity)?
Binding	Protein+	Site(Entity)+
Regulation	Protein/Event, Cause (Protein/Event)?	Site (Entity)?, CSite (Entity)?
Positive_ regulation	Protein/Event, Cause (Protein/Event)?	Site (Entity)?, CSite (Entity)?
Negative_ regulation	Protein/Event, Cause (Protein/Event)?	Site (Entity)?, CSite (Entity)?

requires to indicate the negation and/or speculation based on the extracted and enriched events. We denote that though the original definition of *Binding* events can involve more than two arguments, the number of the *Binding* events with more than two arguments are minor. Most of applications choose to ignore the *Binding* events with more than two arguments for simplicity. For arguments, the names before parentheses are types of arguments while the names in parentheses are the types of the target entities allowed be arguments for this event.

Table B.2 lists the new definition of events in 2013, they are basically the same as the definitions in 2009 except the new event types. Moreover, *Phosphorylation* events can involve CAUSE arguments like the **REG** events. However, in training set and development set, the numbers of examples of new events are less than 10, which are not sufficient for proper training. It is the same for the *Phosphorylation* events that contain CAUSE arguments (18 out of 512 *Phosphorylation* events have CAUSE arguments).

TABLE B.2: BioNLP Genia event extraction task definitions."?" means the number of this argument is 0 or 1, "+" means the number of this argument is at least 1, "*" means the number of this argument is not fixed.

Event Type	Participants	Additional Arguments
Gene_ expression	Theme(Protein)	
Transcription	Theme(Protein)	
Protein_ catabolism	Theme(Protein)	
Localization	Theme(Protein)	Loc(Entity)?
Binding	Theme(Protein)+	Site(Entity)*
Protein_ modification	Theme(Protein), Cause(Protein/Event)?	Site(Entity)?
Phosphorylation	Theme(Protein), Cause(Protein/Event)?	Site(Entity)?
Ubiquitination	Theme(Protein), Cause(Protein/Event)?	Site(Entity)?
Acetylation	Theme(Protein), Cause(Protein/Event)?	Site(Entity)?
Deacetylation	Theme(Protein), Cause(Protein/Event)?	Site(Entity)?
Regulation	Theme(Protein/Event), Cause(Protein/Event)?	Site(Entity)?, CSite(Entity)?
Positive_ regulation	Theme(Protein/Event), Cause(Protein/Event)?	Site(Entity)?, CSite(Entity)?
Negative_ regulation	Theme(Protein/Event), Cause(Protein/Event)?	Site(Entity)?, CSite(Entity)?

B.2 Data statistics

This section lists the statistics of dataset in BioNLP 2009, 2011 and 2013 Genia task reported in [1–3]. Comparing the event numbers listed in Table B.3 and Table B.4, we can see the inconsistency of annotations between BioNLP 2009 and 2011 datasets. Table B.5 shows the number of events in new dataset, where the numbers of examples corresponding to new classes are too small.

TABLE B.3: Data sets in BioNLP 2009 Genia tasks.

	Train	Devel	Test
Abstract	800	150	260
Sentence	74490	1450	2447
Word	176146	33937	57367
Event	8597	1809	3182

Table B.4: Data sets in BioNLP 2011 Genia tasks.

	Training		Development		Test	
	Abs	**Full**	**Abs**	**Full**	**Abs**	**Full**
Articles	800	5	150	5	260	4
Documents	800	108	150	109	150	87
Words	176146	29583	33827	30305	57256	21791
Proteins	9300	2325	2080	2610	3589	1712
Events	8615	1695	1795	1455	3193	1294
Gene_expression	1735	527	356	393	722	280
Transcription	576	91	82	76	137	37
Protein_catabolism	110	0	21	2	14	1
Phosphorylation	169	23	47	64	139	50
Localization	265	16	53	14	174	17
Binding	887	101	249	126	349	153
Regulation	961	152	173	123	292	96
Positive_regulation	2847	538	618	382	987	466
Negative_regulation	1062	247	196	275	379	379

Table B.5: Data sets in BioNLP 2013 Genia tasks.

	Training	Development	Test
Articles	10	10	14
Words	54938	57907	75144
Proteins	3571	4138	4359
Entities	121	314	327
Events	8615	3193	1294
Gene_expression	729	591	619
Transcription	122	98	101
Localization	44	197	99
Protein_catabolism	23	30	14
Binding	195	376	342
Protein_modification	8	1	1
Phosphorylation	117	197	161
Ubiquitination	4	2	30
Acetylation	0	3	0
Deacetylation	0	5	0
Regulation	299	284	299
Positive_regulation	780	883	1144
Negative_regulation	496	532	538

B.3 Ambiguous examples

We listed some ambiguous examples in this section. Figure B.1 shows the example that use stop word "by" as trigger word. Compared to other events, using stop words as trigger word is very hard to recognize especially for the pair-wise relation resolution. Determining the role of "by" in this example requires joint information from both THEME and CAUSE arguments.

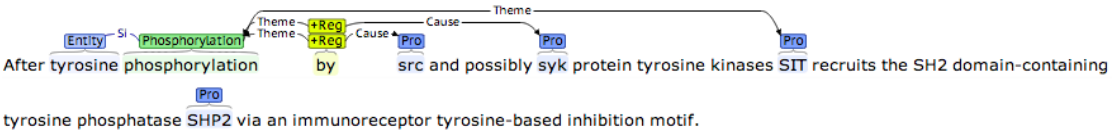
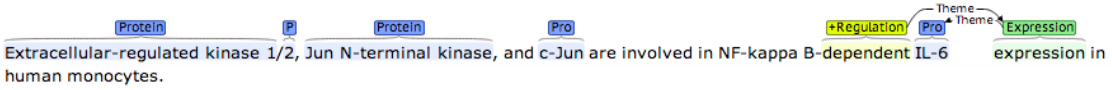
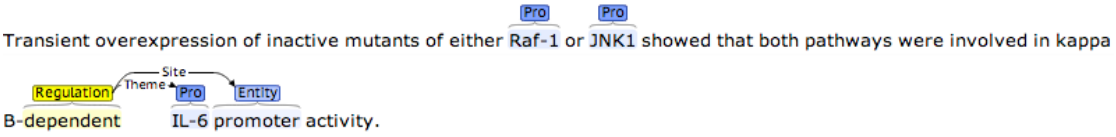


FIGURE B.1: Trigger is stop word "**by**".

Figure B.2 illustrate the examples that contain the trivial word "dependent", which does not have any polarity but is used in both *Regulation* and *Positive_regulation* events. Event the human biomedical expert cannot deduct the polarity of the example given only the sentence without context.



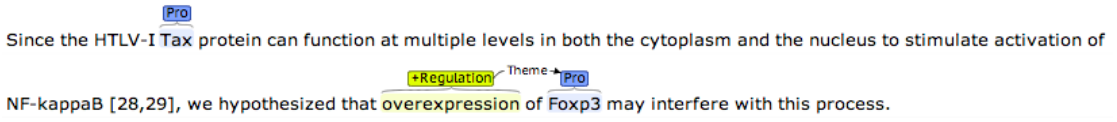
(A) Trivial word as Positive_regulation



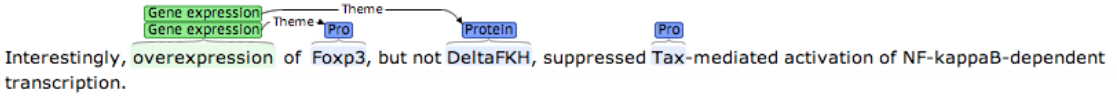
(B) Trivial word as Regulation

FIGURE B.2: Trivial word "**dependent**" in different event classes

Besides the ambiguity between *Regulation* and *Positive_regulation*, the ambiguous words marked both in two irrelative classes are also confusing. Examples presented in Figure B.3 show the ambiguity of word "overexpression", which is marked as two event classes in BioNLP2009 dataset but as only one of the two classes in BioNLP2011 dataset.



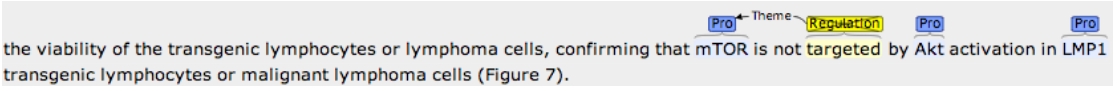
(A) Ambiguous word as Positive_regulation



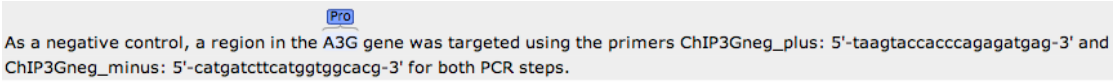
(B) Ambiguous word as Gene_expression

FIGURE B.3: Ambiguous word "**overexpression**" in different event classes

In Figure B.4, we present the examples in the largest misclassification category, which is the misclassification between events and *None*.



(A) Ambiguous example as Positive_regulation



(B) Ambiguous example as false positive prediction

FIGURE B.4: Ambiguous word "**targeted**"

Bibliography

[1] Jin-Dong Kim, Tomoko Ohta, Sampo Pyysalo, Yoshinobu Kano, and Jun'ichi Tsujii. Overview of bionlp'09 shared task on event extraction. In *Proceedings of the BioNLP 2009 Workshop Companion Volume for Shared Task*, pages 1–9, Boulder, Colorado, June 2009. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W09-1401.

[2] Jin-Dong Kim, Yue Wang, Toshihisa Takagi, and Akinori Yonezawa. Overview of genia event task in bionlp shared task 2011. In *Proceedings of BioNLP Shared Task 2011 Workshop*, pages 7–15, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W11-1802.

[3] Jin-Dong Kim, Yue Wang, and Yamamoto Yasunori. The genia event extraction shared task, 2013 edition - overview. In *Proceedings of the BioNLP Shared Task 2013 Workshop*, pages 8–15, Sofia, Bulgaria, August 2013. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W13-2002.

[4] Jung-Jae Kim, Xu Han, Vivian Lee, and Dietrich Rebholz-Schuhmann. Gro task: Populating the gene regulation ontology with events and relations. In *Proceedings of the BioNLP Shared Task 2013 Workshop*, pages 50–57, Sofia, Bulgaria, August 2013. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W13-2007.

[5] Sampo Pyysalo, Tomoko Ohta, and Sophia Ananiadou. Overview of the cancer genetics (cg) task of bionlp shared task 2013. In *Proceedings of the BioNLP Shared Task 2013 Workshop*, pages 58–66, Sofia, Bulgaria, August 2013. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W13-2008.

[6] Tomoko Ohta, Sampo Pyysalo, Rafal Rak, Andrew Rowley, Hong-Woo Chun, Sung-Jae Jung, Sung-Pil Choi, Sophia Ananiadou, and Jun'ichi Tsujii. Overview of the

pathway curation (pc) task of bionlp shared task 2013. In *Proceedings of the BioNLP Shared Task 2013 Workshop*, pages 67–75, Sofia, Bulgaria, August 2013. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W13-2009.

[7] Claire Nédellec, Robert Bossy, Jin-Dong Kim, Jung-Jae Kim, Tomoko Ohta, Sampo Pyysalo, and Pierre Zweigenbaum. Overview of bionlp shared task 2013. In *Proceedings of the BioNLP Shared Task 2013 Workshop*, pages 1–7, Sofia, Bulgaria, August 2013. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W13-2001.

[8] Sampo Pyysalo, Tomoko Ohta, and Jun'ichi Tsujii. Overview of the entity relations (rel) supporting task of bionlp shared task 2011. In *Proceedings of BioNLP Shared Task 2011 Workshop*, pages 83–88, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W11-1812.

[9] Ngan Nguyen, Jin-Dong Kim, and Jun'ichi Tsujii. Overview of bionlp 2011 protein coreference shared task. In *Proceedings of BioNLP Shared Task 2011 Workshop*, pages 74–82, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W11-1811.

[10] Robert Bossy, Julien Jourde, Philippe Bessières, Maarten van de Guchte, and Claire Nédellec. Bionlp shared task 2011 - bacteria biotope. In *Proceedings of BioNLP Shared Task 2011 Workshop*, pages 56–64, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W11-1809.

[11] Sampo Pyysalo, Tomoko Ohta, Rafal Rak, Dan Sullivan, Chunhong Mao, Chunxia Wang, Bruno Sobral, Jun'ichi Tsujii, and Sophia Ananiadou. Overview of the infectious diseases (id) task of bionlp shared task 2011. In *Proceedings of BioNLP Shared Task 2011 Workshop*, pages 26–35, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W11-1804.

[12] Tomoko Ohta, Sampo Pyysalo, and Jun'ichi Tsujii. Overview of the epigenetics and post-translational modifications (epi) task of bionlp shared task 2011. In *Proceedings*

of BioNLP Shared Task 2011 Workshop, pages 16–25, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W11-1803.

[13] Jin-Dong Kim, Sampo Pyysalo, Tomoko Ohta, Robert Bossy, Ngan Nguyen, and Jun'ichi Tsujii. Overview of bionlp shared task 2011. In *Proceedings of BioNLP Shared Task 2011 Workshop*, pages 1–6, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W11-1801.

[14] Sampo Pyysalo, Tomoko Ohta, Jin-Dong Kim, and Jun'ichi Tsujii. Static relations: a piece in the biomedical information extraction puzzle. In *Proceedings of the Workshop on Current Trends in Biomedical Natural Language Processing*, pages 1–9. Association for Computational Linguistics, 2009.

[15] Sampo Pyysalo, Filip Ginter, Juho Heimonen, Jari Björne, Jorma Boberg, Jouni Järvinen, and Tapio Salakoski. Bioinfer: a corpus for information extraction in the biomedical domain. *BMC bioinformatics*, 8(1):50, 2007.

[16] Jin-Dong Kim, Tomoko Ohta, and Jun'ichi Tsujii. Corpus annotation for mining biomedical events from literature. *BMC bioinformatics*, 9(1):10, 2008.

[17] David McClosky, Mihai Surdeanu, and Christopher Manning. Event extraction as dependency parsing for bionlp 2011. In *Proceedings of BioNLP Shared Task 2011 Workshop*, pages 41–45, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W11-1806.

[18] Sebastian Riedel, Hong-Woo Chun, Toshihisa Takagi, and Jun'ichi Tsujii. A markov logic approach to bio-molecular event extraction. In *Proceedings of the BioNLP 2009 Workshop Companion Volume for Shared Task*, pages 41–49, Boulder, Colorado, June 2009. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W09-1406.

[19] Sebastian Riedel and Andrew McCallum. Fast and robust joint models for biomedical event extraction. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1–12, Edinburgh, Scotland, UK., July 2011. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/D11-1001.

[20] Sebastian Riedel and Andrew McCallum. Robust biomedical event extraction with dual decomposition and minimal domain adaptation. In *Proceedings of BioNLP Shared Task 2011 Workshop*, pages 46–50, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W11-1807.

[21] Sebastian Riedel, David McClosky, Mihai Surdeanu, Andrew McCallum, and Christopher D. Manning. Model combination for event extraction in bionlp 2011. In *Proceedings of BioNLP Shared Task 2011 Workshop*, pages 51–55, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W11-1808.

[22] David McClosky, Sebastian Riedel, Mihai Surdeanu, Andrew McCallum, and Christopher D. Manning. Combining joint models for biomedical event extraction. *BMC Bioinformatics*, 13(S-11):S9, 2012.

[23] Jari Björne, Juho Heimonen, Filip Ginter, Antti Airola, Tapio Pahikkala, and Tapio Salakoski. Extracting complex biological events with rich graph-based feature sets. In *Proceedings of the BioNLP 2009 Workshop Companion Volume for Shared Task*, pages 10–18, Boulder, Colorado, June 2009. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W09-1402.

[24] Jari Björne and Tapio Salakoski. Generalizing biomedical event extraction. In *Proceedings of BioNLP Shared Task 2011 Workshop*, pages 183–191, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W11-1828.

[25] Jari Björne and Tapio Salakoski. TEES 2.1: Automated annotation scheme learning in the BioNLP 2013 shared task. In *Proceedings of BioNLP Shared Task 2013 Workshop*, Sofia, Bulgaria, August 2013. Association for Computational Linguistics.

[26] Jin-Dong Kim, Ngan Nguyen, Yue Wang, Jun'ichi Tsujii, Toshihisa Takagi, and Akinori Yonezawa. The genia event and protein coreference tasks of the bionlp shared task 2011. *BMC Bioinformatics*, 13(Suppl 11):S1, 2012. ISSN 1471-2105. doi: 10.1186/1471-2105-13-S11-S1. URL http://www.biomedcentral.com/1471-2105/13/S11/S1.

[27] Yusuke Miyao and Jun'ichi Tsujii. Feature forest models for probabilistic hpsg parsing. *Computational Linguistics*, 34(1):35–80, 2008.

[28] Kenji Sagae and Jun'ichi Tsujii. Dependency parsing and domain adaptation with lr models and parser ensembles. In *EMNLP-CoNLL*, volume 2007, pages 1044–1050, 2007.

[29] Makoto Miwa, Sampo Pyysalo, Tadayoshi Hara, and Jun'ichi Tsujii. A comparative study of syntactic parsers for event extraction. In *Proceedings of the 2010 Workshop on Biomedical Natural Language Processing*, pages 37–45, Uppsala, Sweden, July 2010. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W10-1905.

[30] Chris Quirk, Pallavi Choudhury, Michael Gamon, and Lucy Vanderwende. Msr-nlp entry in bionlp shared task 2011. In *Proceedings of BioNLP Shared Task 2011 Workshop*, pages 155–163, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W11-1825.

[31] Youngjun Kim, Ellen Riloff, and Nathan Gilbert. The taming of reconcile as a biomedical coreference resolver. In *Proceedings of BioNLP Shared Task 2011 Workshop*, pages 89–93, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W11-1813.

[32] Ekaterina Buyko, Erik Faessler, Joachim Wermter, and Udo Hahn. Event extraction from trimmed dependency graphs. In *Proceedings of the BioNLP 2009 Workshop Companion Volume for Shared Task*, pages 19–27, Boulder, Colorado, June 2009. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W09-1403.

[33] Hyoung-Gyu Lee, Han-Cheol Cho, Min-Jeong Kim, Joo-Young Lee, Gumwon Hong, and Hae-Chang Rim. A multi-phase approach to biomedical event extraction. In *Proceedings of the BioNLP 2009 Workshop Companion Volume for Shared Task*, pages 107–110, Boulder, Colorado, June 2009. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W09-1415.

[34] Halil Kilicoglu and Sabine Bergler. Syntactic dependency based heuristics for biological event extraction. In *Proceedings of the BioNLP 2009 Workshop Companion Volume for Shared Task*, pages 119–127, Boulder, Colorado, June 2009. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W09-1418.

[35] Andreas Vlachos and Mark Craven. Biomedical event extraction from abstracts and full papers using search-based structured prediction. In *Proceedings of BioNLP Shared Task 2011 Workshop*, pages 36–40, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W11-1805.

[36] Liam R. McGrath, Kelly Domico, Courtney D. Corley, and Bobbie-Jo Webb-Robertson. Complex biological event extraction from full text using signatures of linguistic and semantic features. In *Proceedings of BioNLP Shared Task 2011 Workshop*, pages 130–137, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W11-1818.

[37] Ehsan Emadzadeh, Azadeh Nikfarjam, and Graciela Gonzalez. Double layered learning for biological event extraction from text. In *Proceedings of BioNLP Shared Task 2011 Workshop*, pages 153–154, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W11-1824.

[38] Sofie Van Landeghem, Jari Björne, Thomas Abeel, Bernard De Baets, Tapio Salakoski, and Yves Van de Peer. Semantically linking molecular entities in literature through entity relationships. *BMC Bioinformatics*, 13(S-11):S6, 2012.

[39] Lishuang Li, Yiwen Wang, and Degen Huang. Improving feature-based biomedical event extraction system by integrating argument information. In *Proceedings of the BioNLP Shared Task 2013 Workshop*, pages 109–115, Sofia, Bulgaria, August 2013. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W13-2015.

[40] Roser Morante, Vincent Van Asch, and Walter Daelemans. A memory-based learning approach to event extraction in biomedical texts. In *Proceedings of the BioNLP 2009 Workshop Companion Volume for Shared Task*, pages 59–67, Boulder, Colorado, June 2009. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W09-1408.

[41] Andrew MacKinlay, David Martinez, and Timothy Baldwin. Biomedical event annotation with crfs and precision grammars. In *Proceedings of the BioNLP 2009 Workshop Companion Volume for Shared Task*, pages 77–85, Boulder, Colorado, June 2009. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W09-1410.

[42] Rune Sætre, Makoto Miwa, Kazuhiro Yoshida, and Jun'ichi Tsujii. From protein-protein interaction to molecular event extraction. In *Proceedings of the BioNLP 2009 Workshop Companion Volume for Shared Task*, pages 103–106, Boulder, Colorado, June 2009. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W09-1414.

[43] Quang Le Minh, Son Nguyen Truong, and Quoc Ho Bao. A pattern approach for biomedical event annotation. In *Proceedings of BioNLP Shared Task 2011 Workshop*, pages 149–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W11-1822.

[44] Kaarel Kaljurand, Gerold Schneider, and Fabio Rinaldi. Uzurich in the bionlp 2009 shared task. In *Proceedings of the BioNLP 2009 Workshop Companion Volume for Shared Task*, pages 28–36, Boulder, Colorado, June 2009. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W09-1404.

[45] Andreas Vlachos, Paula Buttery, Diarmuid Ó Séaghdha, and Ted Briscoe. Biomedical event extraction without training data. In *Proceedings of the BioNLP 2009 Workshop Companion Volume for Shared Task*, pages 37–40, Boulder, Colorado, June 2009. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W09-1405.

[46] Quoc-Chinh Bui and Peter. M.A. Sloot. Extracting biological events from text using simple syntactic patterns. In *Proceedings of BioNLP Shared Task 2011 Workshop*, pages 143–146, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W11-1820.

[47] Quoc-Chinh Bui, David Campos, Erik van Mulligen, and Jan Kors. A fast rule-based approach for biomedical event extraction. In *Proceedings of the BioNLP Shared Task 2013 Workshop*, pages 104–108, Sofia, Bulgaria, August 2013. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W13-2014.

[48] Syed Toufeeq Ahmed, Radhika Nair, Chintan Patel, and Hasan Davulcu. Bioeve: Bio-molecular event extraction from text using semantic classification and dependency parsing. In *Proceedings of the BioNLP 2009 Workshop Companion Volume for Shared Task*, pages 99–102, Boulder, Colorado, June 2009. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W09-1413.

[49] Jari Björne and Tapio Salakoski. Tees 2.1: Automated annotation scheme learning in the bionlp 2013 shared task. In *Proceedings of the BioNLP Shared Task 2013 Workshop*, pages 16–25, Sofia, Bulgaria, August 2013. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W13-2003.

[50] Makoto Miwa, Rune Sætre, Jin-Dong Kim, and Jun'ichi Tsujii. Event extraction with complex event classification using rich features. *Journal of bioinformatics and computational biology*, 8(01):131–146, 2010.

[51] Jörg Hakenberg, Illes Solt, Domonkos Tikk, Luis Tari, Astrid Rheinländer, Nguyen Quang Long, Graciela Gonzalez, and Ulf Leser. Molecular event extraction from link grammar parse trees. In *Proceedings of the BioNLP 2009 Workshop Companion Volume for Shared Task*, pages 86–94, Boulder, Colorado, June 2009. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W09-1411.

[52] Andrew MacKinlay, David Martinez, Antonio Jimeno Yepes, Haibin Liu, W John Wilbur, and Karin Verspoor. Extracting biomedical events and modifications using subgraph matching with noisy training data. In *Proceedings of the BioNLP Shared Task 2013 Workshop*, pages 35–44, Sofia, Bulgaria, August 2013. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W13-2005.

[53] Haibin Liu, Ravikumar Komandur, and Karin Verspoor. From graphs to events: A subgraph matching approach for information extraction from biomedical text. In *Proceedings of BioNLP Shared Task 2011 Workshop*, pages 164–172, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W11-1826.

[54] Sofie Van Landeghem, Yvan Saeys, Bernard De Baets, and Yves Van de Peer. Analyzing text in search of bio-molecular events: a high-precision machine learning framework. In *Proceedings of the BioNLP 2009 Workshop Companion Volume for*

Shared Task, pages 128–136, Boulder, Colorado, June 2009. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W09-1419.

[55] György Móra, Richárd Farkas, György Szarvas, and Zsolt Molnár. Exploring ways beyond the simple supervised learning approach for biological event extraction. In *Proceedings of the BioNLP 2009 Workshop Companion Volume for Shared Task*, pages 137–140, Boulder, Colorado, June 2009. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W09-1420.

[56] Arzucan Ozgur and Dragomir Radev. Supervised classification for extracting biomedical events. In *Proceedings of the BioNLP 2009 Workshop Companion Volume for Shared Task*, pages 111–114, Boulder, Colorado, June 2009. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W09-1416.

[57] Asma Ben Abacha and Pierre Zweigenbaum. Medical entity recognition: A comparison of semantic and statistical methods. In *Proceedings of BioNLP 2011 Workshop*, pages 56–64. Association for Computational Linguistics, 2011.

[58] Kaibo Duan, S. Sathiya Keerthi, Wei Chu, Shirish Krishnaj Shevade, and Aun Neow Poo. Multi-category classification by soft-max combination of binary classifiers. In *In 4th International Workshop on Multiple Classifier Systems*, 2003.

[59] John C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *Advances in Large Margin Classifiers*, pages 61–74. MIT Press, 1999.

[60] D. M. J. Tax and R. P. W. Duin. Using two-class classifiers for multiclass classification. In *Proceedings of the 16th International Conference on Pattern Recognition*, volume 2, pages 124–127 vol.2, 2002.

[61] Kai Hakala, Sofie Van Landeghem, Tapio Salakoski, Yves Van de Peer, and Filip Ginter. EVEX in ST'13: Application of a large-scale text mining resource to event extraction and network construction. In *Proceedings of BioNLP Shared Task 2013 Workshop*, Sofia, Bulgaria, August 2013. Association for Computational Linguistics.

[62] K. Morik, P. Brockhausen, and T. Joachims. Combining statistical learning with a knowledge-based approach - a case study in intensive care monitoring. In *Proceedings of the Sixteenth International Conference on Machine Learning (ICML 1999)*, 1999.

[63] K. Veropoulos, C. Campbell, and N. Cristianini. Controlling the sensitivity of support vector machines. In T. Dean, editor, *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 55–60, 1999.

[64] R. Brent. *Algorithms for minimization without derivatives*. Prentice-Hall, 1973.

[65] Andreas Vlachos and Mark Craven. Biomedical event extraction from abstracts and full papers using search-based structured prediction. *BMC bioinformatics*, 13(Suppl 11):S5, 2012.

[66] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022, 2003.

[67] R. Collobert and J. Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *International Conference on Machine Learning, ICML*, 2008.

[68] Eric H Huang, Richard Socher, Christopher D Manning, and Andrew Y Ng. Improving word representations via global context and multiple word prototypes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 873–882. Association for Computational Linguistics, 2012.

[69] Ciyou Zhu, Richard H Byrd, Peihuang Lu, and Jorge Nocedal. Algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound-constrained optimization. *ACM Transactions on Mathematical Software (TOMS)*, 23(4):550–560, 1997.

[70] John A Nelder and Roger Mead. A simplex method for function minimization. *Computer journal*, 7(4):308–313, 1965.

[71] Margaret H Wright. Direct search methods: Once scorned, now respectable. *Pitman Research Notes in Mathematics Series*, pages 191–208, 1996.

[72] Michael JD Powell. An efficient method for finding the minimum of a function of several variables without calculating derivatives. *The computer journal*, 7(2): 155–162, 1964.

[73] William H Press. *Numerical recipes 3rd edition: The art of scientific computing*. Cambridge university press, 2007.

[74] SJ Wright and J Nocedal. *Numerical optimization*, volume 2. Springer New York, 1999.

[75] Hal Daumé Iii, John Langford, and Daniel Marcu. Search-based structured prediction. *Machine learning*, 75(3):297–325, 2009.

[76] Antoine Bordes, Xavier Glorot, Jason Weston, and Yoshua Bengio. Joint learning of words and meaning representations for open-text semantic parsing. In *International Conference on Artificial Intelligence and Statistics*, pages 127–135, 2012.

[77] Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *Proceedings of NAACL-HLT*, pages 746–751, 2013.

[78] Marco Baroni and Roberto Zamparelli. Nouns are vectors, adjectives are matrices: Representing adjective-noun constructions in semantic space. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1183–1193. Association for Computational Linguistics, 2010.

[79] Dong C Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1-3):503–528, 1989.

[80] William Blacoe and Mirella Lapata. A comparison of vector-based representations for semantic composition. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 546–556. Association for Computational Linguistics, 2012.

[81] Richard Socher, Brody Huval, Christopher D Manning, and Andrew Y Ng. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1201–1211. Association for Computational Linguistics, 2012.

[82] Richard Socher, Jeffrey Pennington, Eric H Huang, Andrew Y Ng, and Christopher D Manning. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 151–161. Association for Computational Linguistics, 2011.

[83] Richard Socher, John Bauer, Christopher D Manning, and Andrew Y Ng. Parsing with compositional vector grammars. In *In Proceedings of the ACL conference*. Citeseer, 2013.

[84] Katsumasa Yoshikawa, Sebastian Riedel, Tsutomu Hirao, Masayuki Asahara, and Yuji Matsumoto. Coreference based event-argument relation extraction on biomedical text. *J. Biomedical Semantics*, 2(S-5):S6, 2011.

[85] Georgiana Dinu, Nghia The Pham, and Marco Baroni. General estimation and evaluation of compositional distributional semantic models. In *Proceedings of the Workshop on Continuous Vector Space Models and their Compositionality*, pages 50–58, Sofia, Bulgaria, August 2013. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W13-3206.

[86] Chang-Ning Huang and Hai Zhao. Chinese word segmentation: A decade review. *Journal of Chinese Information Processing*, 21(3):8–20, 2007.

[87] Nal Kalchbrenner and Phil Blunsom. Recurrent continuous translation models. In *EMNLP*, pages 1700–1709, 2013.