# GENERIC EVENT EXTRACTION USING MARKOV LOGIC NETWORKS

**Zhijie He**

**Bachelor of Engineering**
**Tsinghua University, China**

**A THESIS SUBMITTED**
**FOR THE DEGREE OF MASTER OF SCIENCE**
**SCHOOL OF COMPUTING**
**NATIONAL UNIVERSITY OF SINGAPORE**

**2013**

# Declaration

I hereby declare that this thesis is my original work and it has been written by me in its entirety. I have duly acknowledged all the sources of information which have been used in the thesis. This thesis has also not been submitted for any degree in any university previously.

He Zhijie

Zhijie He

August, 2013

# Acknowledgement

It would not have been possible to write this thesis without the help and support of the kind people around me, to only some of whom it is possible to give particular mention here.

It is with immense gratitude that I acknowledge the support and help of my supervisors Professor Tan Chew Lim, Dr. Jian Su and Dr. Sinno Jialin Pan. Their continuous support constantly led me in the right direction. I would like to thank Professor Tan, who travelled a lot from NUS to I2R to discuss my work with me. I would like to thank Dr. Su and Dr. Pan for their guidance and expertise, which provides me a good direction of my thesis.

I would also thank my colleagues including Man Lan, Qiu Long, Wan Kai, Chen Bin, Zhang Wei, Toh Zhiqiang, Wang Wenting, Tian Shangxuan, and Ding Yang. Without their help this work would have been much harder and taken much longer.

Finally, I am deeply grateful to my parents, for their patient encouragement and support. Their unconditional love gave me courage and enabled me to complete my graduate studies and this research work.

# Contents

# Summary

Event extraction is the extraction of event-related information of interest from text documents. Most of the existing research work splits the event extraction task into three subtasks: event identification, event classification and argument classification. Markov logic networks (MLNs) have been used in bio-molecular event extraction task to minimize the error propagation problem. This application shows limited success. In this thesis, many more features are introduced to enhance the joint inference capability. In addition, the previous study shows that event correlation is useful for event extraction. Thus, we further investigate how to incorporate such inter-sentential information into MLNs to make the information directly interfere with sentence-level inference.

In this thesis, we will first explore extensively the state-of-the-art research of event extraction. Then we will present our framework in MLNs to solve the event extraction task as defined in the Automatic Content Extraction (ACE) Program. Finally, we will demonstrate how to extend our framework from sentence level to document level and how to incorporate document-level features, like event correlation information, into our framework.

We conducted extensive experiments on the ACE 2005 English corpus, to evaluate the generic event extraction scenario. Experimental results show that our system is both efficient and effective in extracting events from text documents. Our framework could make use of the joint learning function provided by MLNs, thus the error propagation problem which is severe and occurs frequently in pipeline systems can be easily avoided. Finally, we have achieved statistically significant improvement after incorporating event correlation information into our framework.

# List of Tables

# List of Figures

# Chapter 1

# Introduction

Nowadays, a tremendous amount of text documents are generated on the Internet, for instance those for news service, media, etc. Unfortunately, without the effort of human beings, these text documents are quite difficult to interpret or analyse. Although time-consuming, extracting critical information from large amount of text sources is one of the key steps towards making better use of this information. If we could automatically extract such information, we could dramatically reduce the human labour and speed up the information extraction process.

In a nutshell, Information Extraction(IE) is a technique to extract structural information from text documents. Generally speaking, IE can be divided into three subtasks, namely entity recognition which identifies entities of interest such as person, location and organization etc; relation extraction identifies the relationship between entities; and event extraction which takes charge of retrieving elements of certain events. In this thesis, we focus on the third task, event extraction, and

particularly event extraction as defined in ACE for the experiment and level of complexity, although the work applies to other event extraction tasks as well.

This chapter will be organized as follows: Section 1.1 will discuss the challenges of this task and state the motivation of this thesis; Section 1.2 will concisely show our contributions; and finally, Section 1.3 will present the outline of entire thesis.

## 1.1 Generic Event Extraction

Event extraction has been extensively researched for a long time. The early stage of investigation into event extraction is a major task called Scenario Template (ST) of the Message Understanding Conference (MUC). The MUC, which began in 1987 and ran until 1998, was sponsored by DARPA for the purpose of fostering research on automatically analysing text information. Since then, many systems (Califf (1998), Soderland (1999), Freitag and Kushmerick (2000), Ciravegna and others (2001), Roth and Yih (2001), Chieu and Ng (2002) etc.) have been developed to extract certain types of events from text documents. In 1999, the Automatic Content Extraction (ACE) programme was developed as a replacement for the MUC. The objective of ACE is to automatically process human language in text from a variety of sources. Lots of research work (Grishman et al. (2005), Ji and Grishman (2008), Liao and Grishman (2010a) etc.) have been dedicated to this task.

In ST task, slots in a given template which is domain dependent will be filled by extracting textual information from text documents. Research on event extraction

has been more complicated than ST. Typically, event extraction is to detect events with event type and corresponding arguments. An example would be as follows:

Ex 1-1 *In 1927 Lisa married William Gresser, a New York lawyer and musicologist.*

A successful event extraction attempt should recognize the event contained in this sentence to be a *Marry* event with *Lisa* as the bride and *William Gresser* as the bridegroom.

There are various applications in event extraction. Event extraction technique can be a useful tool of Knowledge Base Population (KBP) (Ji et al. (2010)). Event extraction technique can extract the relationship between entities and populate an existing knowledge base, which is one of the goals of KBP. Event extraction can be also applied in Question Answering(QA). Events of certain types, such as *Marriage, Be-Born, Attack*, can be used to provide more accurate answers to 5W1H(Who, What, Whom, When, Where and How) questions. Another application which could benefit from event extraction is Text Summarization, which can make use of concepts such as events to represent topics in text documents. Recently, event extraction techniques have been provided in industry. Thomson Reuters, a company providing financial news, launched a web service called Open Calais[1] which can recognize the entities, facts and events in the text.

Event extraction, though a useful task, is extremely challenging. The performances of most of the existing approaches are often too low to be useful for some tasks. Therefore, there are still a lot of issues to be investigated further.

One of the important factors of event extraction is the quality of event corpus.

[1]http://www.opencalais.com/

Building a corpus with high quality is a time-consuming job. Moreover, the more severe problem is that it is difficult for annotators to come to an agreement. Ji and Grishman (2008) showed that the percentage of inter-annotator agreements on event classification is only about 40% on the ACE 05 English corpus. Feng et al. (2012) also showed similar statistical results on the ACE 05 Chinese corpus.

Most of the existing systems(Grishman et al. (2005), Ji and Grishman (2008), Chieu and Ng (2002), Liao and Grishman (2010a)) divide event extraction task into three or more subtasks: trigger identification, event type classification, argument classification, etc. Each of these subtasks is so difficult that many approaches (McClosky et al. (2011), Lu and Roth (2012) etc) which focus on only one subtask have been proposed. Those systems which solve the whole task usually process these subtasks in a pipeline way. However, the main issue of pipeline systems is error propagation, which is more severe in event extraction. To be specific, errors from previous stages could be propagated to the current stage, which is the key factor in lowering the performance of a pipeline system.

Moreover, information within a sentence is sometimes not clear enough to detect an event. For example, the sentence "*He left the company*" may contain a *Transport* event or an *End-Position* event depending on the context. Liao and Grishman (2010a) incorporates event correlation information to help extract events. However, because these constraints involve events in the same document, it is often difficult to incorporate such global constraints into a pipeline system. Therefore, we need a framework that could be easily extended and enriched.

## 1.2 Our Contributions

In this thesis, we propose a unified framework on generic event extraction based on MLNs. Our framework is capable of achieving much higher performance than state-of-the-art sentence level systems. To summarize, we make the following contributions:

- We propose a new unified MLN on generic event extraction. We did extensive experiments to show the performance of our framework. Results show that our framework outperforms the state-of-the-art sentence-level systems.
- Our framework can be easily extended and enriched. To show this, we encode event correlation information into our system. Experimental result show that this information improves the performance of generic event extraction.

## 1.3 Outline of This Thesis

The remainder of this thesis is organized as follows:

Chapter 2 reviews the existing related work. In this chapter, we provide a comprehensive literature review about the different approaches to this task. Since our work is based on MLNs and is inspired from biomedical event extraction, we also give an introduction to MLNs and their application to biomedical event extraction.

Chapter 3 presents our framework on generic event extraction. We first implement the initial framework which is inspired by Riedel (2008). Then we add some crucial features to the initial framework to make our framework perform better.

Chapter 4 describes our attempt to incorporate event correlation information to our framework. This chapter gives a comprehensive trial to show that it is quite easy to extend and enrich our framework.

Chapter 5 presents the experimental evaluation. We did extensive experiments on the ACE 05 English corpus, which showed that our framework can improve the performance of event extraction. In this chapter, we give a detailed discussion and analysis of our experimental results.

Chapter 6 concludes our research presented in this thesis and provides several possible future research directions.

# Chapter 2

# Literature Review

Event extraction has been actively studied in recent years. Many approaches have been developed to extract events from text documents.

This chapter will first review several existing approaches used in event extraction systems. These approaches will be categorized into two categories, namely *rule induction approaches*, and *machine-learning-based approaches*. We will then conduct a detailed review of a novel branch of machine learning technique, i.e. *Markov logic networks (MLNs)* used in bio-molecular event extraction.

## 2.1 Rule Induction approaches

Events can be captured by rules which can either be learnt from data or hand-crafted by domain experts. To this end, many distinct rule learning algorithms (Califf (1998), Soderland (1999), Freitag and Kushmerick (2000), Ciravegna and

others (2001), Roth and Yih (2001)) have been proposed. Shallow features in Natural Language Processing (NLP) and active learning methods are adopted by some of these approaches and have been shown to be effective.

RAPIER(Califf (1998)) induced pattern-matched rules to extract fillers for the slots when given a template. For this purpose, an inductive logic programming technique was employed to learn rules for pre-fillers, fillers and post-fillers respectively. Such technique is a compression-based search approach starting from specific to general cases. First of all, the most specific rules for each slot in the template are used for each training example. Then it iteratively compacts all the rules by replacing these rules with more general ones and removing the old rules that are subsumed by the new ones. As for features, RAPIER used tokens, part-of-speech tags and semantic class information.

WHISK(Soderland (1999)) used an active learning method to learn template rules which are in the form of regular expressions. This method repeatedly adds new training instances which are almost missing during the training procedure. Then it discards rules with errors on the new instance and generates new rules for the slots which are not covered by the current rules. As for features, WHISK also used tokens and semantic class information.

Boosted Wrapper Induction (BWI) (Freitag and Kushmerick (2000)) learned a large number of simple rules and combined them using boosting. It learns rules for start tags and end tags in separate models and then uses a histogram of field lengths to estimate the probability of the length of a fragment. As for features, BWI used the tokens, and lexical knowledge(obtained using gazetteers) such as first names, last names etc.

$LP^2$(Ciravegna and others (2001)), a rule based system, induced symbolic rules for identifying start and end tags. Like BWI, it identifies start and end tags separately. It also learns rules to correct tags labelled by certain rules. Like RAPIER and BWI, $LP^2$ also used a bottom up search approach in its learning algorithm. In addition to features like tokens and orthographic features such as lowercase, captalizations etc, $LP^2$ used some shallow NLP features such as morphology, part-of-speech tag and a gazetteer.

Systems based on rule induction approaches have a number of desirable properties. Firstly, it is easy to read and understand the rules learnt by rule induction systems. Thus the issues occurred in rule induction systems often can be solved by inspecting the learnt rules. Moreover, a rule often has a natural first order version. Thus techniques for learning first-order rules also can be readily used in rule induction.

The major problem with rule induction approaches is that the rule learning algorithms often scale relatively poorly with the sample size, particularly on noisy data. Another problem in rule induction learning systems is that it is difficult to select a number of good seed instances to start the rule induction process. Much research can be done towards this field.

## 2.2 Machine-Learning-Based Approaches

Machine learning techniques have been employed widely in many natural language processing tasks. This section will review several approaches which are based on supervised learning.

Chieu and Ng (2002) used a maximum entropy model to do the template filling task. Based on their model, they constructed a three-stage pipeline system. The first stage is to identify whether a document contains events or not. If the document contains at least one event, the entities in this document will be further classified for each slot in the second stage. Note that in this stage, only relevant types of entities are classified. For example, to fill in the corporate name slot, they would only classify the organization entities. In the final stage, for each pair of entities, a classifier will be used to identify whether these two entities are in the same event or not. They used syntactic features provided by BADGER(Fisher et al. (1995)) and semantic class information as features of their model.

ELIE(Finn and Kushmerick (2004)) is a two tier template filling system. Like Chieu and Ng (2002), ELIE treated the information extraction task as a kind of classification problem whose goal is to classify each token into one of the classes of start-slot, end-slot or none. ELIE used support vector machines to induce a set of two-level classifiers. The purpose of the classifiers of the first level is to achieve high precision, while that of the classifiers of the second level is to achieve high recall.

Grishman(Grishman et al. (2005)) built a novel sentence-level baseline system for the ACE 2005 event extraction task. Their approach combines the rule-based approach and statistical learning approach. Rules are automatically learnt from the training set and then applied to find the potential triggers and arguments, both of which will be further classified by some statistical classifier. The features used in this system were syntactic features such as part-of-speech tag, dependency and semantic class information.

Ahn (2006) developed a pipeline event extraction system on the ACE 2005 corpus, in which the event extraction task is divided into two stages: *trigger classification* and *argument classification*. In the trigger classification stage, tokens will be categorized into one of the 34 predefined classes(33 *event* types and one *none* type). In the argument classification stage, entities will be characterized into one of the 36 predefined classes(35 *argument* types and one *none* type) given the classified triggers in the previous stage. The major difference between this and Chieu and Ng (2002)'s work is that Ahn (2006) put additional efforts in identifying triggers of certain events.

ACE event extraction confines the event mentions to within one sentence. However utilizing only sentence-level information is not enough in some scenarios because of the ambiguity of natural language. Consider, in an article, such a sentence: *Tom leaves the company*. If what the article wants to express is that Tom is no longer an employee of this company, then we can consider the event contained in this sentence to be an *End-Position* event. However, if what the article wants to express is that Tom departs from the company, then we can consider the event contained in this sentence to be a *Transport* event. Researchers have tried to utilize global features such as document level information, event correlation and entity background information to obtain higher performance for event extraction.

Ji and Grishman (2008) proposed to incorporate global evidence from a cluster of related documents to refine local decisions. They developed a system based on the work of Grishman et al. (2005). In the testing procedure, in addition to performing sentence level event extraction, they performed document-level event extraction by using information retrieval technique to retrieve related documents

as a cluster given a potential trigger and arguments. To achieve consistency, they adjusted the trigger and the arguments according to some predefined rules. Basically, these rules remove the triggers and arguments with low confidence in local sentence or cluster, and set the confidence of the trigger and arguments to the higher one between local sentence and cluster. Compared with the work of Grishman et al. (2005), the system performance is considerably increased by the global information.

Liao and Grishman (2010a) presented an approach to add event correlation information to boost the performance. The motivation of this idea is quite intuitive: in articles, events are often correlated with each other. An *Attack* event for instance, often leads to an *Injure* or *Die* event. Besides, the arguments are often correlated as well, since they often have some relationship in their corresponding correlated events. For example, the *Target* in an *Attack* event may be the *Victim* of an *Injure* event. To incorporate event correlation information, the researchers developed a two-phase system. The first phase is the same as what was done in Grishman et al. (2005). Then two argument level classifiers are trained in the second phase: trigger classifier and argument classifier. The former is to retag the low confidence triggers filtered out from the first phase. And the latter classifier is to retag entities with low confidence in the same sentence of the tagged triggers.

Hong et al. (2011) claimed that the background information of the entity could provide useful information to help extract events. Statistical results show that entities having the same background often participate in similar events as one same role. To collect background information about the entities, a search engine is used to query each entity and related documents are collected to determine the

entity's background. However, this approach is not good enough for practical use, since the result sets of the search engine query may change and we do not know whether the query result is semantically related to the entity or not.

McClosky et al. (2011) presented an interesting event extraction approach by using dependency parsing. In the training process, they converted the triggers and arguments of events into dependency trees and generated a reranking dependency parser. In the testing process, they first recognized the triggers in the sentence, and then used the trained dependency parser to parse the sentence into an event structure with the argument type as the label of the edge from trigger to entity. Instead of outputting the best dependency tree, they output top-n dependency trees and used a reranker to rerank the trees to get the best event structures.

Liao and Grishman (2011a) acquired topic information to help event extraction. They proposed that events are often related to specific topics. For example, a document whose topic is *war* is more likely to contain *Attack* or *Injure* events. They compared an unsupervised topic model with a multi-label supervised topic model. Results show that the unsupervised approach performs better.

Other methods such as active learning(Liao and Grishman (2011b)) and boostrapping (Liao and Grishman (2010b), Huang and Riloff (2012)) which are widely used in other related tasks in the NLP domain, were also tested in event extraction task.

Supervised approaches for event extraction can take advantage of state-of-the-art machine learning techniques, since adding features to a supervised model is more straight-forward.

Event extraction is a challenging task and unsupervised methods are much

more challenging than supervised methods. Despite the challenges, the benefits of unsupervised methods are more attractive. For instance, unsupervised methods avoid the situation where substantial human efforts are needed to annotate the training instances required in the supervised methods. As we know, human annotations can be very expensive and sometimes impractical. Even if annotators are available, getting annotators to agree with each other is often a difficult task. Worse still, annotations often can not be reused: experimenting on a different domain or dataset typically requires annotating new training instances for that particular domain or dataset.

Lu and Roth (2012) performed event extraction by using semi-Markov conditional random fields. Their work identifies event arguments, assuming that the correct event type is given. Besides the supervised approach, they also investigated an unsupervised approach by incorporating predefined patterns into their model to do event extraction. Six patterns were predefined for matching arguments. The model prefers an argument set that well matches to the patterns. The key step for this approach is to define patterns as accurately as possible, and thus domain experts are needed. The researchers show that the unsupervised approach almost catches up with the supervised approach in some specific event types.

In summary, machine-learning-based approaches have been widely used in the event extraction task. Most of these systems are sentence level systems which take a sentence as input. A wider scope of features such as topics of documents, event correlation, entity correlation etc., is used to enhance the performance. The event extraction task is often split into subtasks like event identification, event classification and argument classification and solves these subtasks in a pipeline

way. Though unsupervised learning for the event extraction task is more attractive, its performance is much lower than that of supervised learning. Furthermore, event extraction only extracts specific types of events, and thus supervised learning is more effective.

## 2.3 Bio-molecular Event Extraction via Markov Logic Networks

This section conducts a detailed review of the Markov logic networks and its application in bio-molecular event extraction.

### 2.3.1 Markov Logic Networks

Markov logic networks (MLNs) (Richardson and Domingos (2006)) combine markov networks and first order logic. An MLN L consists of a set of weighted first-order logic formulas $\{(\phi_i, w_i)\}$, where $\phi_i$ is a first order logic formula and $w_i$ is the weight of the formula. When binding the free variables in the formulas by constants, it defines a markov network with one node per ground atom and one feature per ground formula. The weight of the feature is the weight of the corresponding ground formula. Then we can define a distribution over sets of ground atoms or so-called possible worlds. The probability of a possible world $\mathbf{y}$ is defined as follows:

$$p(\mathbf{y}) = \frac{1}{Z} \exp\left( \sum_{(\phi_i, w_i) \in L} w_i \sum_{\mathbf{c} \in C^{\phi_i}} f_{\mathbf{c}}^{\phi_i}(\mathbf{y}) \right) \tag{2.1}$$

Here $\mathbf{c}$ is one possible binding of the free variables to constants in $\phi_i$ and $C^{\phi_i}$ is the set of all possible bindings of the free variables in $\phi_i$. $f_{\mathbf{c}}^{\phi_i}$ is a ground formula representing a binary feature function. It will return 1 if the ground formula we get by replacing the free variables in $\phi_i$ with the constants in $\mathbf{c}$ is true, and 0 otherwise. $Z$ is a normalization constant. The above distribution corresponds to a markov network whose nodes represent ground atoms and factors represent ground formulas.

As in first-order logic, each formula is constructed from predicates using logical connectives and quantifiers. Take the following formula as an example:

$$(\phi_i, w_i) : word(a, b) \Rightarrow event(a) \tag{2.2}$$

The above formula indicates that if token a is word b, then token a is an event. As stated before, formula 2.2 cannot be violated in first-order logic, while it can be violated with some probability in MLNs. Here a and b are free variables which can be replaced by constants, and *word* and *event* are evidence predicate and hidden predicate respectively. Evidence predicates are those whose values can be known from given observations, while hidden predicates are the target predicates whose values need to be predicted. From this example, we can see that *word* is an evidence predicate because we can check whether token a is word b or not. *Event* is hidden predicate since this is something we would like to predict.

This thesis uses the inference and learning algorithms provided in the open source *thebeast*[1] package. In particular, we employed the maximum a posteriori (MAP) inference and the 1-best Margin Infused Relaxed Algorithm (MIRA) (Crammer

[1]https://code.google.com/p/thebeast/

and Singer (2003)) online learning method.

Given an MLN L and a set of observed grounding atoms $x$, a set of hidden ground atoms $\hat{y}$ with maximum a posteriori probability is to be inferred

$$\hat{\mathbf{y}} = \arg\max_{\mathbf{y}} p(\mathbf{y}|\mathbf{x}) = \arg\max_{\mathbf{y}} s(\mathbf{y}, \mathbf{x})$$

where

$$s(\mathbf{y}, \mathbf{x}) = \sum_{(\phi_i, w_i) \in L} w_i \sum_{\mathbf{c} \in C^{\phi_i}} f_{\mathbf{c}}^{\phi_i}(\mathbf{y}, \mathbf{x})$$

can be considered as the score that evaluates the goodness of solution $(\mathbf{y}, \mathbf{x})$. The MAP inference in *thebeast* package is implemented by Integer Linear Programming(ILP). A detailed introduction to transforming the MAP inference to an ILP problem can be found in Riedel (2008).

For weight learning, the online learning method 1-best MIRA learns the weights which separate the gold solution from all the wrong solutions with a large margin. This can be achieved by solving the quadratic program as follows:

$$\begin{aligned} &\min ||\mathbf{w}_t - \mathbf{w}_{t-1}|| \\ s.t.\ &s(\mathbf{y}_i, \mathbf{x}_i) - s(\mathbf{y}', \mathbf{x}_i) \geq L(\mathbf{y}_i, \mathbf{y}') \\ &\forall (\mathbf{x}_i, \mathbf{y}_i) \in D \text{ and } \mathbf{y}' = \arg\max_{\mathbf{y}} s(\mathbf{y}, \mathbf{x}_i | \mathbf{w}_{t-1}) \end{aligned}$$

Here D is the training instances and $t$ is the number of iterations, $s(\mathbf{y}, \mathbf{x}|\mathbf{w})$ is the score of solution $(\mathbf{y}, \mathbf{x})$ given a weight $\mathbf{w}$. We try to find a new weight $\mathbf{w}_t$ which can guarantee that the difference between the gold solution $(\mathbf{y}_i, \mathbf{x}_i)$ and the best solution $(\mathbf{y}', \mathbf{x}_i)$ is at least as big as the loss $L(\mathbf{y}_i, \mathbf{y}')$, while changing the old weight

$\mathbf{w}_{t-1}$ as little as possible. The loss function $L(\mathbf{y}_t, \mathbf{y}')$ is the number of false positive and false negative ground atoms for all hidden atoms.

### 2.3.2 Bio-molecular Event Extraction using MLNs

MLNs have been successfully applied to bio-molecular event extraction. Here we will review an approach which uses MLNs to do bio-molecular event extraction.

Bio-molecular event extraction is the main concern of the BioNLP 09 Shared Task. This task focuses on the extraction of bio-molecular events, particularly on proteins. There are 9 types of bio-events to be extracted. The core task involves event trigger and primary argument. One of the major differences between ACE events and bio-molecular events is that the arguments of bio-molecular events could be events, while arguments are only limited to entities, values and time expressions in ACE events.

Riedel (2008) first used MLNs to extract bio-molecular events. Their system achieved 4th place on the core task in the competition, but still lagged about 8% behind the 1st place system. They designed a hidden predicate for each target, such as trigger identification, trigger classification etc, and found some global constraints to help joint inference. With the help of MLNs, they could bypass the need to design and implement specific inference and training methods. As we will see later in Chapter 3 and Chapter 5, a new MLN which is inspired by Riedel (2008) will be proposed and proved to have a good performance on generic event extraction[2]

[2]Though Poon and Vanderwende (2010), which also is an MLN framework to extract bio-molecular events, outperformed Riedel (2008) about 5% in F-Score, they defined some context specific formulas. The framework presented in Riedel (2008) is more general so we believed that it is a good point to start from.

# Chapter 3

# Generic Event Extraction Framework via MLNs

In this chapter, a unified event extraction framework is presented to resolve generic event extraction. This framework is based on Markov logic networks (MLNs), which have been introduced in Chapter 2.

This chapter is organized into three major sections. We start with the problem description and the definitions of predicates in Sections 3.1 and 3.2. Then we introduce a base MLN framework, which is inspired from bio-molecular event extraction, in Section 3.3. Finally, we present a full MLN framework for generic event extraction in Section 3.4.

## 3.1 Problem Statement

Ideally, given a text document, an event extraction system should identify all the event mentions in the document with their corresponding types and arguments, if they have any. To be specific, we take a sentence and corresponding entity information as inputs. Then the goals are:

- Event identification: identify triggers within the input sentence if it has any.
- Event classification: assign an event type with the trigger identified.
- Argument classification: for each event, assign an argument type for each entity in the sentence if the entity is an argument for the event.

With results of the three goals, we can output events and their arguments from the input sentence. Take the following sentence as an example:

> Ex 3-1 *In the West Bank, an eight-year-old Palestinian boy as well as his brother and sister were wounded late Wednesday by Israeli gunfire in a village north of the town of Ramallah.*

In the above sentence, we can extract out an *Attack* event as shown in Table 3.1.

## 3.2 Predicates

Before discussing the framework, some predicates must first be defined, because these predicates are the foundation of complex features which can be expressed in

| Trigger | gunfire |
|---|---|
| | |
| **Argument Type** | **Value** |
| Attacker | Israeli |
| Target | an eight-year-old Palestinian boy |
| Target | his brother |
| Target | sister |
| Place | a village north of the town of Ramallah |
| Time | late Wednesday |

Table 3.1: An Event Example

the form of first order logic formulas.

### 3.2.1 Hidden Predicates

Hidden predicates are predicates whose truth values are to be predicted in our framework. They are similar to the labels to be predicted in other discriminative models like support vector machines.

We define three hidden predicates corresponding to the goals mentioned in Section 3.1: *event(tid)* for event identification; *eventtype(tid, e)* for event classification; *argument(tid, eid, r)* for argument classification. Table 3.2 shows descriptions of the above hidden predicates.

| **Predicate** | **Description** |
|---|---|
| event(tid) | The token whose index is *tid* triggers an event. |
| eventtype(tid, e) | The token whose index is *tid* triggers an event whose type is *e*. |
| argument(tid, eid, r) | The entity whose identifier is *eid* is an argument of type *r* for the event triggered by the token whose index is *tid*. |

Table 3.2: Hidden Predicates

Recall that most of the event extraction systems are pipeline systems where triggers will be identified first, then event types will be classified, and finally positive events will be assigned with arguments. In MLNs, however, we can accomplish these three goals simultaneously. As discussed before, in a pipeline system, the major problem is error propagation. The errors that occur in the previous stages cannot be corrected in the current stage. In event extraction systems, this problem is much more critical, since the performance of each stage is not high. However, in MLNs, the objectives can be solved simultaneously. In addition, with the global constraints, the final results of these three objectives would be in a consistent state. Thus, we could avoid error propagation in our framework.

### 3.2.2 Evidence Predicates

Evidence predicates, as fundamental features, provide information which can be observed before inference. Therefore, evidence predicates are used in the condition

part of formulas.

| **Predicate** | **Description** |
|---|---|
| word(tid, w) | The token tid is word w. |
| lemma(tid, s) | The lemma of token tid is s. |
| pos(tid, p) | The part-of-speech tag of the token tid is p. |
| dep(i, j, d) | The token i is head of the token j with dependency d according to Stanford Dependency Parser. |
| path(i, j, p) | Labelled dependency path p between token i and token j. |
| pathnl(i, j, p) | Unlabelled dependency path p between token i and token j. |
| entity(eid, hid, s, e, n) | Entity eid, which starts from token s and ends at token e, has type n and its head word is token hid. |
| dict(tid, e, prec) | Token tid triggers an event whose type is e with prior estimate prec in training data. |
| allowed(e, n, r) | Entity n is allowed to play argument r in event e. |

Table 3.3: Evidence Predicates

The evidence predicates used here are listed in Table 3.3. The *word*, *lemma* and *pos* predicates deliver syntactic information of tokens. Since the *argument* predicate is to predict the relationship between an entity and a token, we need *dep*, *path* and *pathnl* predicates to relate tokens with relation information. Figure 3.1 shows an example explaining what the *path* and *pathnl* predicates mean. We use the Stanford Parser (De Marneffe et al. (2006)) to generate dependencies for the sentence shown in Figure 3.1. The dependency path between the token "*Center*" and the token "*deaths*" is a path starting from token "*Center*", going through

token "*recorded*" and ending at token "*deaths*". So the labelled dependency path is *path*(4, 7, "nsubj←dobj→"), and the dependency path without labels is *pathnl*(4, 7, "←→"). Here the arrows represent the direction of the dependency edge.

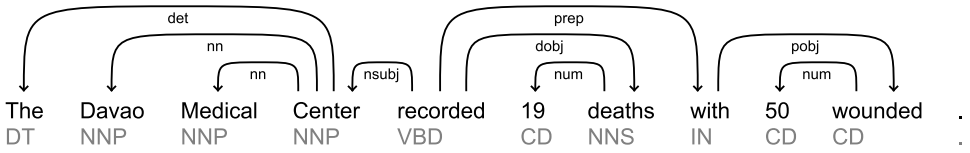


Figure 3.1: An Example to Illustrate *path* and *pathnl* Predicates

Furthermore, information about entities within the input sentence is necessary, since entities will play as arguments in events. Here the *entity* predicate represents an entity. The head word of an entity is the token with maximum height within the span of the entity. For example, "*The Davao Medical Center*" is an entity in the sentence shown in Figure 3.1. The head word of this entity is token "*Center*". Moreover, since the head word cannot represent an entity, we use an identifier to represent an entity.

We also define a predicate named *dict* to collect all the triggers with their corresponding event types in the training data. The *prec* term provides the prior estimate of how likely it is to trigger a corresponding event. We calculate the *prec* term for predicate *dict(i, e, prec)* as follows:

$$prec = \exp\left(\frac{N_e}{N_i} \times \frac{N_e}{N_{ie}}\right) = \exp\left(\frac{N_e^2}{N_i \cdot N_{ie}}\right) \tag{3.1}$$

where $N_e$ is the number of events of type $e$ that are triggered by token i in the training data; $N_i$ is the number of occurrences of token i; and $N_{ie}$ is the number of events of all types that are triggered by token i.

Each argument type only allows a specific set of entities to fill in. For instance, only an entity whose type is *Person* could be a *Victim* argument for an *Injure* event. In order not to assign an entity with an impossible argument type for an event, we define the *allowed* predicate.

## 3.3 A Base MLN

In this section, we will present a base MLN for generic event extraction, which is inspired by Riedel (2008). To be specific, we will describe formulas for *event*, *eventtype* and *argument* respectively.

### 3.3.1 Local Formulas for Event Predicate

A formula is local if it relates any number of evidence predicates to exactly one hidden predicate.

First of all, we add formula 3.2. The weight of this formula indicates how likely a token is to be an event trigger, which is called a bias feature.

$$event(i) \tag{3.2}$$

Note that the term $i$ in formula 3.2 is a free variable, it can be bound by the constants of its domain. Given a sentence, all the indices of the tokens in the sentence can be assigned to the term $i$.

Then a set of formulas which are so called "bag-of-words" features is added:

$$P(i, +t) \Rightarrow event(i) \tag{3.3}$$

where $P \in \{word, lemma, pos\}$. Note that the "+" notation means that for each possible combination of constants whose corresponding variables are with prefix "+" there is a separate weight for the corresponding formula. So a formula with variables preceding "+" will generate many formulas by replacing those variables preceding "+" with constants. For example, when P is the *word* predicate, and there are two constants {"*go*", "*home*"} of word, then the following formulas will be generated:

$$word(i, \text{“}go\text{”}) \Rightarrow event(i) \tag{3.4}$$

$$word(i, \text{“}home\text{”}) \Rightarrow event(i) \tag{3.5}$$

A higher weight indicates that the word will trigger an event with the higher probability. Thus, the weight associated with formula 3.4 will be higher than that associated with formula 3.5. This is because the word "*go*" often indicates an *Transport* event, while the word "*home*" does not trigger any event.

Next, we add the following formula

$$dep(h, i, d) \wedge word(h, +w) \Rightarrow event(i) \tag{3.6}$$

The operator $\wedge$ in formula 3.6 is the logical AND operator. The term $h$ is the index of a token in the sentence and the term $d$ is the dependency label between token

$i$ and token $h$. The above formula captures context information around a trigger. For example, if the word "*go*" has a dependency with the word "*home*", then it is very likely that the word "*go*" is a trigger.

The above formulas were inspired by MLNs for bio-molecular event extraction (BioMLN). As the experimental results will show, BioMLN is not capable of doing well in generic event extraction. As a result, we have to add more formulas which are more suitable for generic event extraction.

A dictionary is helpful in providing domain information, and therefore we collect the triggers and their corresponding events in the training data as a dictionary. To facilitate this information, we add the following formulas:

$$dict(i, e, prec) \wedge P(i, +t) \Rightarrow event(i) \tag{3.7}$$

where $P \in \{word, lemma, pos\}$. The *dict* predicate in these formulas can narrow the scope of the formula, so the weight will be more accurate. These formulas will capture information about how likely it is that the token will trigger an event in the testing data if the token triggers an event in the training data. The term *prec* in *dict* predicate here will multiply the weight of each constant corresponding to term $t$ in $P$ predicate. With this form, we can incorporate probabilities and other numeric quantities like prior estimate in a principled fashion.

In English, phrases are often used to express an action or describe an event. For example, "*go home*" often indicates a *Transport* event. This feature is often referred to as a n-gram feature in many NLP tasks. Here we add one formula to

capture the bigram feature.

$$lemma(i, +t1) \wedge lemma(i+1, +t2) \Rightarrow event(i) \tag{3.8}$$

Trigram is not necessary since it is very sparse and does not make much sense. We only use the *lemma* predicate here, since we want to ignore the tense of the phrase.

Finally, a formula which is similar to formula 3.2 is added:

$$dict(i, +e, prec) \Rightarrow event(i) \tag{3.9}$$

For each token in the dictionary, the probability of triggering an event is different. The above formula estimates how likely a token which is in the dictionary is to be a trigger given that it will trigger an event $e$ with a prior estimation *prec*.

### 3.3.2 Local Formulas for Eventtype Predicate

First of all, we reuse all the aforementioned formulas that are applied to *event* by only replacing the *event* predicate with *eventtype*, as shown in Table 3.4. Recall that the first three formulas were all inspired by BioMLN. Besides, we also propose three new formulas specially designed for event identification, as shown in the last three rows of Table 3.4.

| |
|---|
| $eventtype(i, +e)$ |
| $P(i, +t) \Rightarrow eventtype(i, +e)$ where $P \in \{word, lemma, pos\}$ |
| $dep(h, i, d) \wedge word(h, +w) \Rightarrow eventtype(i, +e)$ |
| $dict(i, +e, prec) \wedge P(i, +t) \Rightarrow eventtype(i, e)$ where $P \in \{word, lemma, pos\}$ |
| $lemma(i, +t1) \wedge lemma(i + 1, +t2) \Rightarrow eventtype(i, +e)$ |
| $dict(i, +e, prec) \Rightarrow eventtype(i, e)$ |

Table 3.4: Part of Local Formulas for Eventtype Predicate

Event classification has to predict each token into one of the predefined types (including a type corresponding to "*not an event*"), which is much more complicated than event identification. Thus, we have to add more features for the *eventtype* predicate.

Some types of events were found to be correlated with some kinds of entities. For instance, a sentence containing an entity whose type is "*Exploding*" often contains an "*Attack*" event. The following formula expresses this situation:

$$dict(i, +e, prec) \wedge entity(id, h, a, b, +n) \Rightarrow eventtype(i, e) \tag{3.10}$$

$$dict(i, +e, prec) \wedge entity(id, h, a, b, +n) \wedge lemma(i, +s) \Rightarrow eventtype(i, e) \tag{3.11}$$

Formula 3.11 captures the feature that each trigger may have a specific pattern in combining different entity types for different events.

The dependency relation between the trigger and the entity will help a lot in classifying event type. For instance, in the sentence "*He was killed*", the

dependency relation between the word "*He*" and the word "*killed*" is "*nsubpass*", which means that the word "*He*" is the passive subject of the word "*killed*". This information will increase the probability of correctly identifying "*killed*" as an *Attack* event. Thus the following formula was added:

$$dep(h, i, +d) \wedge entity(id, h, a, b, +n) \Rightarrow eventtype(i, +e) \tag{3.12}$$

Finally we add a formula to generate bias for dependencies.

$$dep(i, h, +d) \Rightarrow eventtype(i, e) \tag{3.13}$$

### 3.3.3 Local Formulas for Argument Predicate

While the *event* predicate and the *eventtype* predicate are tagged for each token, the *argument* predicate is link prediction, which is to predict the label for the relationship between a token and an entity. We add four categories of local formulas for the *argument* predicate.

The first category of formulas is about lexical and syntactic features. We relate dependency features (*dep*, *path*, *pathnl*) with other features like *word*, *lemma* and *entity*. Dependency features define the relationship between two tokens, which is helpful for predicting the label of *argument* predicate. Note that only the first two formulas come from BioMLN.

| |
|---|
| $P(i,j,+p) \wedge entity(k,j,a,b,e) \Rightarrow argument(i,k,+r)$ where $P \in \{dep, path, pathnl\}$ |
| $P(i,j,+p) \wedge T(i,+t) \wedge entity(k,j,a,b,e) \Rightarrow argument(i,k,+r)$ where $P \in \{dep, path, pathnl\}$ and $T \in \{word, lemma, pos\}$ |
| $dict(i,e,prec) \wedge entity(id,h,a,b,n) \wedge dep(i,h,+d) \wedge P(i,+s) \Rightarrow argument(i,id,+r)$ where $P \in \{word, lemma, pos\}$ |
| $dict(i,e,prec) \wedge entity(id,h,a,b,n) \wedge path(i,h,+p) \wedge P(i,+s) \Rightarrow argument(i,id,+r)$ where $P \in \{word, lemma, pos\}$ |
| $dict(i,e,prec) \wedge entity(id,h,a,b,n) \wedge pathnl(i,h,+p) \wedge P(i,+s) \Rightarrow argument(i,id,+r)$ where $P \in \{word, lemma, pos\}$ |
| $dict(i,e,prec) \wedge entity(id,h,a,b,+n) \wedge P(i,h,+p) \Rightarrow argument(i,id,+r)$ where $P \in \{dep, path, pathnl\}$ |
| $entity(id,h,a,b,n) \wedge dep(i,h,+d) \Rightarrow argument(i,id,+r)$ |
| $entity(id,h,a,b,+n) \wedge P(i,h,+p) \Rightarrow argument(i,id,+r)$ where $P \in \{dep, path, pathnl\}$ |
| $dict(i,e,prec) \wedge entity(id,h,a,b,n) \wedge dep(i,h,+d) \Rightarrow argument(i,id,+r)$ |

Table 3.5: Lexical and Syntactic Features

The next category of formulas to be added is distance and position features. Note that *distance*(h-i) in the formulas is a function which will return the difference between h and i as an integer. The first formula captures that the word before an entity often leaks some information about what type of argument it will be. For example, the phrase *at home* often indicates that the entity *home* will be an argument whose type is *Place*. The remaining formulas of this category incorporate the distance information of entities. Usually, there are some patterns for distance between a trigger and an entity. For example, in "*John married Lily*", the entity following *married* is usually an argument of the *Marry* event.

| |
|---|
| $dict(i, e, prec) \wedge entity(id, h, a, b, n) \wedge P(h-1, +t) \Rightarrow argument(i, id, +r)$ where $P \in \{word, lemma, pos\}$ |
| $dict(i, e, prec) \wedge entity(id, h, a, b, n) \wedge +distance(a-i) \Rightarrow argument(i, id, r)$ |
| $dict(i, e, prec) \wedge entity(id, h, a, b, n) \wedge word(i, +w) \wedge +distance(a-i) \Rightarrow argument(i, id, +r)$ where $P \in \{word, lemma\}$ |
| $dict(i, e, prec) \wedge entity(id, h, a, b, +n) \wedge +distance(a-i) \Rightarrow argument(i, id, +r)$ |

Table 3.6: Position and Distance Features

For the third category, we also add formulas to capture bias for each observed predicate in Table 3.7. These formulas will serve as the prior estimation for the various predicates. Note that only the first formula comes from BioMLN.

| |
|---|
| $argument(i, k, +r)$ |
| $dict(i, e, prec) \wedge entity(id, h, a, b, n) \wedge P(i, +t) \Rightarrow argument(i, id, r)$, where $P \in \{word, lemma, pos\}$ |
| $dict(i, e, prec) \wedge entity(id, h, a, b, +n) \Rightarrow argument(i, id, r)$ |
| $dict(i, +e, prec) \wedge entity(id, h, a, b, n) \Rightarrow argument(i, id, r)$ |
| $dict(i, e, prec) \wedge entity(id, h, a, b, n) \wedge P(i, h, +t) \Rightarrow argument(i, id, r)$, where $P \in \{dep, path, pathnl\}$ |

Table 3.7: Bias Features

The last category of formulas is to investigate the help of word correlation, event and argument correlation etc features. The first formula in Table 3.8 tries to capture the pattern between the potential trigger and the word preceding the head word of the entity. Entity correlation information may be helpful in predicting argument type. For example, in the ACE event extraction task, entities such as *Exploding* and *Shooting* often correlate to *Attack* events. Thus, we add the second and the third formulas. Finally, different events usually contain different kinds of arguments, and the last formula will learn this pattern.

| |
|---|
| $dict(i, e, prec) \wedge entity(id, h, a, b, n) \wedge lemma(i, +w1) \wedge lemma(h-1, +w2) \Rightarrow argument(i, id, +r)$ |
| $dict(i, e, prec) \wedge entity(id1, h1, a1, b1, +n1) \wedge entity(id2, h2, a2, b2, +n2) \wedge id1 \neq id2 \Rightarrow argument(i, id, +r)$ |
| $dict(i, e, prec) \wedge entity(id, h, a, b, +n) \Rightarrow argument(i, id, +r)$ |
| $dict(i, +e, prec) \wedge entity(id, h, a, b, n) \Rightarrow argument(i, id, +r)$ |

Table 3.8: Misc Features

## 3.4 A Full MLN

In this section, a full MLN is described. Our full MLN includes a set of global formulas in additional to all the formulas described in the base MLN.

A formula is global if it involves more than two hidden predicates. There are two kinds of global formulas. One is hard global formulas whose weight is infinite, the other is soft global formulas whose weight can be learned. The hard global formulas is a hard constraint that cannot be violated. In this full MLN, a set of hard global formulas is added to increase the performance of all the three goals described in Section 3.1.

Global formulas play a key role in implementing joint learning. In the base MLN, because all the local formulas only involve one hidden predicate, the solutions to the three goals are independent. Therefore, there may be inconsistent solutions. For example, $event(i)$ is true for a token whose index is $i$, but $eventtype(i, e)$ is false for every event types. Since global formulas relate to more than two hidden predicates, when one hidden predicate is predicted confidently, it will propagate the confidence to the other hidden predicate in the same global formula. So in this full MLN, the solutions to the three goals are consistent.

The global formulas are shown in Table 3.9. The first six formulas were inspired by BioMLN.

| **Formula** | **Description** |
|---|---|
| $event(i) \Rightarrow \exists e \ s.t. \ eventtype(i, e)$ | If token i is a trigger, then it must has an event type. |
| $eventtype(i, e) \Rightarrow event(i)$ | If token i triggers an event, then it must be a trigger. |
| $argument(i, id, r) \Rightarrow event(i)$ | If token i has a argument, then it must be a trigger. |
| $eventtype(i, e1) \wedge e1 \neq e2 \Rightarrow \neg eventtype(i, e2)$ | Only one event type can a token trigger. |
| $argument(i, id, r1) \wedge r1 \neq r2 \Rightarrow \neg argument(i, id, r2)$ | An entity can only play **one** argument for an event. |
| $eventtype(i, e) \wedge entity(id, h, a, b, n) \wedge \neg allowed(e, n, r)$ $\Rightarrow \neg argument(i, id, r)$ | The argument an entity plays should be allowed according to guideline. |
| $entity(id, h, a, b, n) \Rightarrow \neg event(h)$ | The head word of an entity should not be a trigger. |
| $entity(id1, h1, a1, b1, n1) \wedge entity(id2, h2, a2, b2, n2)$ $\wedge dep(hid1, hid2, \text{“}conj\text{”}) \wedge argument(tid, hid1, r1)$ $\Rightarrow argument(tid, hid2, r2)$ | If the head words of two entities are connected with “*conj*” dependency, and one of them is an argument of an event, then the other one is also an argument of the event. |

Table 3.9: Global Formulas

All the global formulas are hard constraints, which means that they cannot be violated. The first four formulas tell us that we can only assign one event type for a potential trigger. Note that we don't have the constraint that every event should have at least an argument, since there are events without any argument. The next two formulas restrict the number of roles each entity can play for an event to be one, and the argument that the entity playing should be allowed, for example, *Person* entity can not be argument *Place* for an *Attack* event. By inspecting the training corpora, we find that if two entities are connected by “*conj*” dependency, which occurs when they are connected by conjunction such as “*and*” and “*or*”, then the two entities often play the same argument for an event. Thus, we added the last formula to capture this pattern.

# Chapter 4

# Encoding Event Correlation for Event Extraction

One of the advantages of Markov logic networks (MLNs) is its expressiveness. Because of this, we can easily extend our sentence level framework to document level. This chapter shows how to extend our framework to document level and incorporate event correlation information into it.

## 4.1 Motivation

Liao and Grishman (2010a) proposed using cross event information to improve the performance of event extraction. Roughly speaking, cross event information consists of event correlation information and argument correlation information. Here is an example for event correlation: events like *Attack* often lead to *Injure* or

*Die* events. Figure 4.1 shows the co-occurrence frequency of *Injure*, *Attack* and *Die* with the 33 event types(including itself) in the ACE 05 English corpus. We can see that only a few events such as *Injure, Attack, Meet, Die* and *Transport* that have frequently occurred together with the *Attack* event. For the argument correlation, here is an example: the *Target* of an *Attack* event in the same document probably be the *Victim* of the *Injure* event.

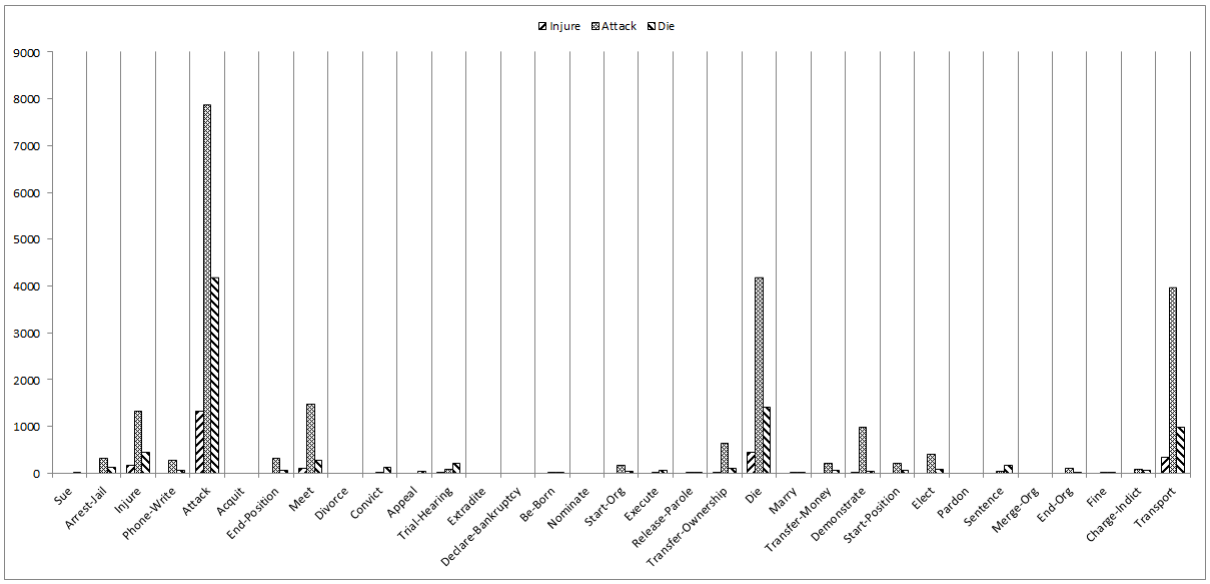


Figure 4.1: Co-occurrence of a certain event type with the 33 ACE event types (Here only Injure, Attack, Die are involved as examples)

Here we show an example[1] of how event correlation could help event classification.

Ex 4-1 *British Chancellor of the Exchequer Gordon Brown on Tuesday* ***named*** *the current head of the country's energy regulator as the new chairman ...* ***Former*** *senior banker Callum McCarthy* ***begins*** *what is one of the most important jobs ... when incumbent Howard Davies* ***steps*** *down. Davies is* ***leaving*** *to* ***become*** *chairman ... As well as* ***previously*** *holding senior positions at ... McCarthy was* ***formerly*** *a top civil servant at the Department of Trade and*

[1]in the ACE 05 English corpus whose file index is AFP_ENG_20030401.0476

*Industry ...*

In Ex 4-1, our sentence level system can find events like *Nominate* event (triggered by *"named"*), *End-Position* events (triggered by *"Former"*, *"steps"*, *"formerly"*), and *Start-Position* events (triggered by *"begins"*, *"become"*). The triggers of these events are easier to detect because they have more specific meanings. Though *"steps"* has multiple meanings, which is more difficult to identify, the phrase *"steps down"* makes it easier to be identified. The trigger *"leaving"* also triggers an *End-Position* event, but our system cannot correctly tag it. This is because *"leaving"* does not always trigger *End-Position* event in training corpora. If we just look at the sentence, we may tag it as a *Transport* event since local context does not provide enough information. With event correlation information, we can tag it as *End-Position* event since most of the events in this document are *End-Position* and *Start-Position* events.

As mentioned in Chapter 2, Liao and Grishman (2010a) presented a system with two stages to facilitate cross event information. They first used a baseline system to extract events. Then the events with high confidence would be the input of the second phase to infer correlated events.

Though Liao and Grishman (2010a) has proposed a system using cross event information, their system has some drawbacks. Firstly, error propagation problem is severe in their system. This is because that the F-score of event classification is not high enough. Secondly, they donot evaluate events without arguments.

One of advantages of MLNs is joint learning. With joint learning, error propagation can be avoided. Thus, it is natural to implement cross event in MLNs. As we will see

later, the complexity of MLNs will increase exponentially when adding soft global formulas. For simplicity, in this thesis, we encode part of cross event information into our MLN. To be specific, event correlation information is encoded in our MLN, while the argument correlation information is to be handled in the future work.

## 4.2 Event Correlation Information In MLN

Chapter 3 presented a sentence-level framework. In this framework, it is difficult to incorporate document level information such as event correlation. Since this information involves events in other sentences, when we are processing one sentence, we can not facilitate information from other sentences. In order to use this kind of information, we extend our framework to document level.

In the sentence-level framework, each sentence is treated as an instance, while in the document-level framework in this chapter, each document is treated as an instance. For each predicate except *allowed* predicate, we add a term *sid*. For instance, we change *word(i, w)* into *word(sid, i, w)*. In this way, we are able to make use of the information of other sentences when we predict the current sentence.

Basically, the cross event information is about the correlation between every pair of events. MLNs are good at modelling relationship features which can easily incorporate the cross-event idea. We can just add one simple formula to implement this idea:

$$eventtype(sid1, tid1, +e1) \wedge eventtype(sid2, tid2, +e2) \tag{4.1}$$

This formula means that we would like to learn different weights for different pairs of events.

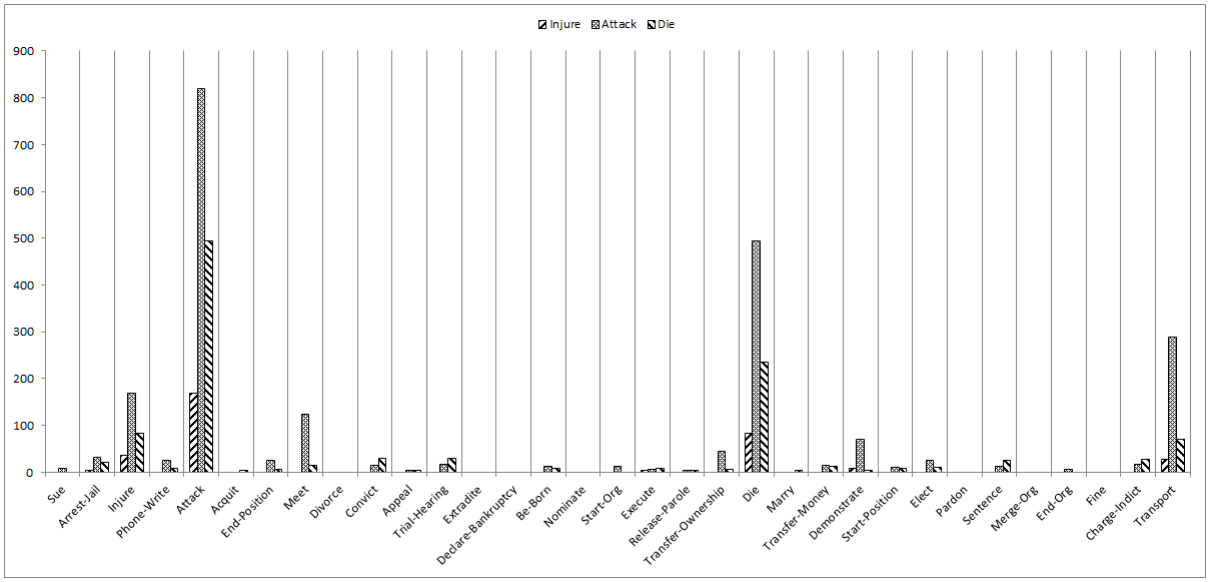


Figure 4.2: Co-occurrence of a certain event type with the 33 ACE event types within next sentence (Here only Injure, Attack, Die are involved as examples)

However, if we immediately use the above version of the formula, the problem space is too large to solve. This is because that for each combination of a token pair and a event type pair, there is one grounding formula corresponding to formula 4.1. When this formula is applied to the whole document, there would be $n^2e^2$ grounding formulas, where $n$ is the number of tokens in the document and $e$ is the number of event types we would like to predict. Therefore, formula 4.1 will increase the space complexity and time complexity. One way to reduce the problem space is to narrow the context. Here we assume that two consecutive sentences are in the same context. Thus, the number of grounding formulas would be $l^2e^2$, where $l$ is

the number of tokens in the two consecutive sentences. Since $l$ is much smaller than $n$, the grounding formulas will be reduced a lot. Therefore, the problem space will be much smaller than before. Figure 4.2 shows that the correlations between events still exist under this condition. When this figure is compared with Figure 4.1, we can see that some weak correlations are filtered. After filtering weak correlations, the weights of event pairs which are really correlated will be more accurate. Thus, we refine the formula in the following way:

$$eventtype(sid, tid1, +e1) \wedge eventtype(sid, tid2, +e2) \tag{4.2}$$

$$eventtype(sid, tid1, +e1) \wedge eventtype(sid + 1, tid2, +e2) \tag{4.3}$$

In the above formulas, we try to learn the relationship between every pair of events in two consecutive sentences.

# Chapter 5

# Experimental Evaluation

This chapter evaluates the performance of our framework and shows the experimental results. An extensive experimental study is conducted to show the performance of our framework. Our framework is evaluated on the ACE 05 English corpus, a comprehensive description of which will be presented first. Following this, the experimental setup is described. Finally the experimental results and discussion are presented. The results show that our framework outperforms the state-of-the-art sentence level system.

## 5.1 ACE Event Extraction Task Description

In this thesis, all of the experiments are reported on the ACE 05 English corpus. Thus, we will describe the ACE event extraction task in this section.

### 5.1.1 ACE Terminology

First of all, we will describe some basic terminologies related to the ACE Extraction Task to facilitate our understanding of the ACE event extraction task.

**Entity** An ACE entity is an object or a set of objects in one of the semantic categories of interest. An entity may have more than one entity mentions. ACE 05 entities have three attributes: type, subtype, and class. For the event extraction task, we only use the subtype attribute. The types and subtypes are listed in Table 5.1.

| **Type** | **Subtypes** |
|---|---|
| FAC (Facility) | Airport, Building-Grounds, Path, Plant, Subarea-Facility |
| GPE (Geo-Political Entity) | Continent, County-or-District, GPE-Cluster, Nation, Population-Center, Special, State-or-Province |
| LOC (Location) | Address, Boundary, Celestial, Land-Region-Natural, Region-General, Region-International, Water-Body |
| ORG (Organization) | Commercial, Educational, Entertainment, Government, Media, Medical-Science, Non-Governmental, Religious, Sports |
| PER (Person) | Group, Indeterminate, Individual |
| VEH (Vehicle) | Air, Land, Subarea-Vehicle, Underspecified, Water |
| WEA (Weapon) | Biological, Blunt, Chemical, Exploding, Nuclear, Projectile, Sharp, Shooting, Underspecified |

Table 5.1: ACE 05 Entity Types and Subtypes

**Entity Mention** An entity mention is the extent of text that refers to an entity. In ACE annotation, a reference such as a pronoun to an entity is also annotated as an entity mention.

**Value** An ACE value is a quantity which has semantic meaning of interest. There are 5 types of values in ACE 05: Contact-Info, Numeric, Crime, Job-Title, Sentence. The Contact-Info class can be further divided into E-Mail, Phone-Number and URL subtypes. Also, the Numeric class has two subtypes: Money and Percent. The other 3 types of values do not have subtypes. A value could be an argument of an event.

**Value Mention** A value mention is the extent of text that refers to a value.

**Timex2** An ACE Timex2 is a time expression. A Timex2 can also be an argument of an event.

**Timex2 Mention** the extent of text that refers to a Timex2.

**Event** An event indicates that a state change incidence occurs. An ACE event is a structural record which contains one trigger and zero or more arguments. The arguments could be entities, values and time expressions. An ACE event often contains one or more event mentions. Table 5.2 shows the ACE 05 event types and subtypes. Besides event types and subtypes, the ACE 05 corpus also annotates other attributes like *modality, polarity, genericity* and *tense*. This thesis will only focus on subtype attribute tagging, by using which, type attributes can be easily inferred. Thus, when we mention event types, we are referring to the event subtypes here and after.

| **Types** | **Subtype** |
|---|---|
| Life | Be-Born, Marry, Divorce, Injure, Die |
| Movement | Transport |
| Transaction | Transfer-Ownership, Transfer-Money |
| Business | Start-Org, Merge-Org, Declare-Bankruptcy, End-Org |
| Conflict | Attack, Demonstrate |
| Contact | Meet, Phone-Write |
| Personnel | Start-Position, End-Position, Nominate, Elect |
| Justice | Arrest-Jail, Release-Parole, Trial-Hearing, Sue, Charge-Indict, , Convict, Sentence, Fine, Execute, Extradite, Acquit, Appeal, Pardon |

Table 5.2: ACE 05 Event Types and Subtypes

**Event Mention** An ACE event mention is a sentence or phrase that mentions an event, and the extent of the event mention is defined to be the whole sentence within which the event is mentioned.

**Event Mention Trigger** A trigger of an event mention is the word that most clearly expresses that event. Every event mention is indicated by a trigger.

**Event Mention Argument** An argument of an event is a mention with some relationship with that event. The mention could be an entity mention, a value mention or a timex2 mention. An argument can also be referred to as a role. Table 5.3 shows all the argument types.

| | | | | |
|---|---|---|---|---|
| Person | Place | Time-Within | Time-Starting | Time-Ending |
| Time-Before | Time-After | Time-Holds | Time-At-Beginning | Time-At-End |
| Agent | Victim | Instrument | Artifact | Vehicle |
| Price | Origin | Destination | Buyer | Seller |
| Beneficiary | Giver | Recipient | Money | Org |
| Attacker | Target | Entity | Position | Crime |
| Defendant | Prosecutor | Adjudicator | Plaintiff | Sentence |

Table 5.3: Argument Types defined by ACE 05

### 5.1.2 ACE Event Mention Detection task

The ACE Event Mention Detection task(VMD) requires that certain specified types of events that are mentioned in the document be detected and that triggers and arguments of these events should be recognized and merged into a unified representation for each detected event. Generally speaking, an event extraction system should include two sub-tasks: VMD and event coreference. In this thesis, we will only deal with the VMD task, whereas event coreference handling will be left to be our future work.

Here is an example of event mention detection and recognition:

> Ex 5-1 *Kelly, the US assistant secretary for East Asia and Pacific Affairs,* ***arrived*** *in Seoul from Beijing Friday.*

In (Ex 5-1), entity mentions are listed in Table 5.4. This sentence contains a *Transport* event which is triggered by the word "*arrived*". Table 5.5 shows the arguments of this event. The possible entity types list the types of entities that the corresponding arguments can take. And the entity mention ID is the entity mention which is the value of the corresponding argument.

| Entity Mention ID | Head Word | Entity Type | Entity Subtype |
|---|---|---|---|
| 001 | Kelly | PER | Individual |
| 002 | Seoul | GPE | Population-Center |
| 003 | Beijing | GPE | Population-Center |
| 004 | Friday | Timex2 | |

Table 5.4: Entity Mentions in Ex 5-1

| Argument | Possible Entity Types | Entity Mention ID |
|---|---|---|
| Destination | GPE, LOC, FAC | 001 |
| Origin | GPE, LOC, FAC | 003 |
| Artifact | PER, WEA, VEH | 001 |
| Time-Within | Timex2 | 004 |

Table 5.5: Arguments in Ex 5-1

## 5.2 Experimental Setup

This section presents our experimental setup. introducing the experimental platform, dataset, and evaluation metric. Finally, we will describe how to preprocess the corpus.

### 5.2.1 Experimental Platform

The experiments were conducted using *thebeast* software, which is freely available for research purpose. All the experiments were done on in Ubuntu 12.04 with JDK

1.6. Our system was powered with a 4-core Intel Core i5 3.20GHZ CPU and 4GB memory.

### 5.2.2 Dataset

We used the ACE 2005 English corpus as our dataset. There are 599 English documents in this dataset. We followed Liao and Grishman (2010a)'s evaluation, randomly selecting 40 documents as our testing set, and using the rest of the documents (559 documents) as training data. We randomly generated 5 testing sets in the experiment.

The ACE English documents are divided into 6 portions. Table 5.6 shows the word count and file count for each portion. There are four versions of the data in the ACE 05 English corpus. Each version corresponds to one annotating process. In our experiments, we use the final version of the data which has the highest quality and has the time expressions normalized.

| **Portion** | **Words** | **Files** |
|---|---|---|
| Newswire | 48399 | 106 |
| Broadcast News | 55967 | 226 |
| Broadcast Conversations | 40415 | 60 |
| Weblog | 37897 | 119 |
| Usenet | 37366 | 49 |
| Conversational Telephone Speech | 34868 | 39 |
| Total | 259889 | 599 |

Table 5.6: ACE English Corpus Statistics

Table 5.7 shows the number of samples of different event types. We can see that the distribution of event types is not uniform. The *Attack* event occurs twice as often as the *Transport* event, however, the *Pardon* event only occurs twice. Therefore, events like *Pardon* do not have enough samples for training. This is one of the reasons the performance of event extraction is low in the ACE 05 English corpus. The distribution of argument mentions is shown in Table 5.8. Similar to the distribution of events, the numbers of some arguments like *Price* and *Time-At-End* are too low to be learnt.

| **Event Type** | **Count** | **Event Type** | **Count** |
|---|---|---|---|
| Attack | 1542 | Demonstrate | 81 |
| Transport | 721 | Sue | 76 |
| Die | 598 | Convict | 76 |
| Meet | 280 | Be-Born | 50 |
| End-Position | 212 | Start-Org | 47 |
| Transfer-Money | 198 | Release-Parole | 47 |
| Elect | 183 | Appeal | 43 |
| Injure | 142 | Declare-Bankruptcy | 43 |
| Transfer-Ownership | 127 | End-Org | 37 |
| Phone-Write | 123 | Divorce | 29 |
| Start-Position | 118 | Fine | 28 |
| Trial-Hearing | 109 | Execute | 21 |
| Charge-Indict | 106 | Merge-Org | 14 |
| Sentence | 99 | Nominate | 12 |
| Arrest-Jail | 88 | Extradite | 7 |
| Marry | 83 | Acquit | 6 |
| | | Pardon | 2 |

Table 5.7: Event Mentions Statistics

| **Argument Type** | **Count** | **Argument Type** | **Count** |
|---|---|---|---|
| Place | 1124 | Org | 124 |
| Entity | 881 | Buyer | 104 |
| Time-Within | 849 | Adjudicator | 103 |
| Artifact | 738 | Money | 88 |
| Attacker | 707 | Vehicle | 86 |
| Person | 699 | Plaintiff | 84 |
| Victim | 673 | Time-Holds | 78 |
| Destination | 571 | Sentence | 78 |
| Target | 518 | Time-Starting | 61 |
| Agent | 430 | Seller | 45 |
| Defendant | 378 | Beneficiary | 32 |
| Instrument | 308 | Time-Before | 30 |
| Crime | 260 | Time-After | 27 |
| Origin | 191 | Prosecutor | 27 |
| Recipient | 151 | Time-Ending | 24 |
| Position | 140 | Time-At-Beginning | 20 |
| Giver | 136 | Time-At-End | 16 |
| | | Price | 12 |

Table 5.8: Argument Mentions Statistics

Unlike with Liao and Grishman (2010a)'s evaluation which only randomly selected 40 documents in the Newswire portion for testing, we randomly selected 40 documents from all the six portions. We want to implement a framework that could be able to extract events regardless of the source of the document. Besides, our framework does not have to tune parameters, so there is no need to split a development set.

### 5.2.3 Evaluation Metric

In the ACE Evaluation Plan(eva (2005)), the scores of every slot of the event were combined into a final score. This score was not intuitive since we do not know how well the system extracts events and arguments. To look into the details of our system and compare the results with other approaches, we followed Ji and Grishman (2008)'s evaluation method. We wanted to evaluate the system performance at three levels, i.e. event identification, event classification and argument classification. The event identification tells us how well the system can detect events. The event classification tells us how well the system can extract events and their types. The argument classification tells us how well the system can find and fill roles for the extracted events.

We use the precision, recall and F-Score to evaluate the system performance. These metrics are widely used in pattern recognition tasks. They are defined as follows:

$$Precision = \frac{|System\ samples \bigcap Key\ samples|}{|System\ samples|}$$
$$Recall = \frac{|System\ samples \bigcap Key\ samples|}{|Key\ samples|}$$
$$F - Score = \frac{2 * Precision * Recall}{Precision + Recall}$$

We also define how two samples are matched with respect to the following metric:

| **Evaluation Metric** | **Matched Elements** |
|---|---|
| Event identification | Trigger start offset<br>Trigger end offset |
| Event Classification | Event type and subtype<br>Trigger start offset<br>Trigger end offset |
| Argument Classification | Event type and subtype<br>Argument head start offset<br>Argument head end offset<br>Argument role |

Table 5.9: The elements that need to be matched for each evaluation metric

### 5.2.4 Preprocessing Corpora

Before generating ground atoms for each predicate, we have to preprocess the documents. First of all, we use the Stanford Parser[1] to parse the documents. After parsing, we can get sentences of the documents, the part-of-speech tags of tokens, and the dependency relationships between tokens. Also, we use the lemmatizer provided in the Stanford Parser to lemmatize the tokens. Then we collect the triggers in the training set as a dictionary. Finally we generate the dependency path between tokens.

[1]http://nlp.stanford.edu/software/corenlp.shtml

## 5.3 Results and Analysis

### 5.3.1 NYU Baseline

We used a state-of-the-art English event extraction system from Grishman et al. (2005) as our baseline. This system is built on top of the JET(Java Extraction Toolkit)[2], which is freely available for research purposes.

This system is a pipeline system which combines pattern matching with statistical models. In the training process, a set of patterns is automatically constructed for each event mention in the corpus. Then all the inaccurate patterns are filtered out. Finally, a set of maximum entropy based classifiers are trained: an argument classifier which is to detect the arguments, a role classifier which is to classify types of arguments, and a trigger classifier which is to identify events.

In the testing process, they first apply patterns to match the potential events and arguments. Then the argument classifier will try to detect more arguments from the rest of the entity mentions in the same sentence. If some arguments can be found in this step, a role classifier is used to assign roles for the arguments. And finally, the trigger classifier will be applied to determine whether this potential event is reportable or not.

We use this system to reproduce a baseline result for event extraction given gold entity mentions. The baseline result is shown in Table 5.10. Note that the worst and optimum are determined in terms of the F-Score of argument classification.

[2]http://cs.nyu.edu/grishman/jet/license.html

| | event identification | | | event classification | | | argument classification | | |
|---|---|---|---|---|---|---|---|---|---|
| | P | R | F | P | R | F | P | R | F |
| worst | 0.628 | 0.549 | 0.586 | 0.610 | 0.534 | 0.570 | 0.331 | 0.337 | 0.334 |
| optimum | 0.656 | 0.473 | 0.550 | 0.632 | 0.456 | 0.530 | 0.416 | 0.332 | 0.369 |
| average | 0.637 | 0.529 | 0.578 | 0.615 | 0.511 | 0.558 | 0.365 | 0.336 | 0.349 |

Table 5.10: NYU Baseline

### 5.3.2 BioMLN Baseline

This section describes the construction of an MLN, which was directly borrowed from Riedel (2008), to produce a baseline named BioMLN. The formulas of BioMLN are shown in Chapter 3. Note that the formulas of BioMLN include local formulas and global formulas.

The performance of the BioMLN is shown in Table 5.11. Compared with the NYU baseline, the performance of event identification is almost the same as that of the NYU baseline in terms of the F-score. The performance of event classification is a little higher; however, the performance of argument classification is much lower than the NYU baseline.

It may be observed that the recall of argument classification in BioMLN is fairly low, which means that the system can not retrieve correct arguments effectively. Since BioMLN is directly borrowed from Riedel (2008) which is designed for bio-molecular event extraction, we can infer that generic events are much more complex than bio-molecular events. For bio-molecular events, the number of argument types

is much smaller than for ACE events. Besides, the bio-molecular event extraction task is to extract events from bio-medical literature which is fairly well written text. On the contrary, the ACE 05 English corpus consists of various types of text such as broadcast, conversation, weblog etc. Basically, the bio-molecular event extraction task and the generic event extraction task are in different domains, so it is not surprising that the performance of argument classification is fairly low.

Furthermore, the *argument* predicate corresponds to the argument classification part. To predict the *argument* predicate, we have to predict the trigger and assign arguments. Though there are global formulas to constrain the relation between event and arguments, without efficient features, we can not improve the performance of argument classification. Thus, we have to define a new framework for generic event extraction.

| | event identification | | | event classification | | | argument classification | | |
|---|---|---|---|---|---|---|---|---|---|
| | P | R | F | P | R | F | P | R | F |
| worst | 0.648 | 0.476 | 0.549 | 0.634 | 0.466 | 0.537 | 0.341 | 0.051 | 0.089 |
| optimum | 0.716 | 0.552 | 0.623 | 0.699 | 0.539 | 0.608 | 0.459 | 0.088 | 0.148 |
| average | 0.662 | 0.514 | 0.578 | 0.645 | 0.501 | 0.563 | 0.429 | 0.073 | 0.125 |

Table 5.11: Results of BioMLN

### 5.3.3 Results of Base MLN

Table 5.12 shows the results of our base MLN. First of all, compared with the NYU baseline, our base MLN gains 1.2% improvement for event identification, and 4.6% for event classification, which is a good improvement. However, for the argument

classification, our base MLN lags by about 4% in terms of the F-score.

From the presented results, we can see that after adding more formulas which are suitable for the ACE event extraction, we can improve the F-score of event identification by 1.2%, event classification by 4.1%, and argument classification by 18.2% compared with the BioMLN. The 18.2% improvement verifies that the formulas defined for bio-molecular event extraction can not be directly applied to generic event extraction.

| | event identification | | | event classification | | | argument classification | | |
|---|---|---|---|---|---|---|---|---|---|
| | P | R | F | P | R | F | P | R | F |
| worst | 0.542 | 0.717 | 0.617 | 0.529 | 0.655 | 0.586 | 0.338 | 0.220 | 0.266 |
| optimum | 0.576 | 0.704 | 0.634 | 0.584 | 0.667 | 0.623 | 0.456 | 0.273 | 0.341 |
| average | 0.521 | 0.680 | 0.590 | 0.563 | 0.652 | 0.604 | 0.395 | 0.251 | 0.307 |

Table 5.12: Results of Base MLN

### 5.3.4 Results of Full MLN

Table 5.13 shows the results of the full MLN. Compared with the results of the base MLN, the full MLN increases about 7% in F-score for event identification, 3.5% for event classification, and about 9% for argument classification. One of the advantages of MLNs is joint learning. Without joint learning, we can not structurally predict the event structure.

Figure 5.1 shows the comparison of the various systems above in terms of average F score. For the event identification, the average F-score of the base MLN is almost

the same with that of the NYU baseline system. However, the NYU baseline system lags by about 5% in terms of the F-score when compared with the base MLN. This is because the error propagates from event identification to event classification in the NYU baseline system. While in the base MLN, event identification and event classification are independent, so the F-score of these two goals are almost the same. The performance is further improved in our full MLN due to the benefit of joint learning provided by the hard global formulas. Thus, Compared with the NYU sentence level baseline system which is state-of-the-art on the ACE 05 English corpus, our full MLN outperforms the NYU sentence-level baseline system.

| | event identification | | | event classification | | | argument classification | | |
|---|---|---|---|---|---|---|---|---|---|
| | P | R | F | P | R | F | P | R | F |
| worst | 0.661 | 0.700 | 0.680 | 0.619 | 0.655 | 0.637 | 0.463 | 0.317 | 0.376 |
| optimum | 0.694 | 0.670 | 0.682 | 0.671 | 0.648 | 0.659 | 0.548 | 0.346 | 0.424 |
| average | 0.672 | 0.654 | 0.663 | 0.649 | 0.631 | 0.639 | 0.537 | 0.315 | 0.396 |

Table 5.13: Results of Full MLN

### 5.3.5 Adding Event Correlation Information

Here we show the result of adding the event correlation information described in Chapter 4. Compared with the result of the full MLN shown in Table 5.13, there is about 1% improvement for event identification, a 1.1% improvement for event classification, and about 1% improvement for argument classification (all based on t-test with confidence $> 95\%$). Though the event correlation information is added to the *eventtype* predicate, event identification and argument classification

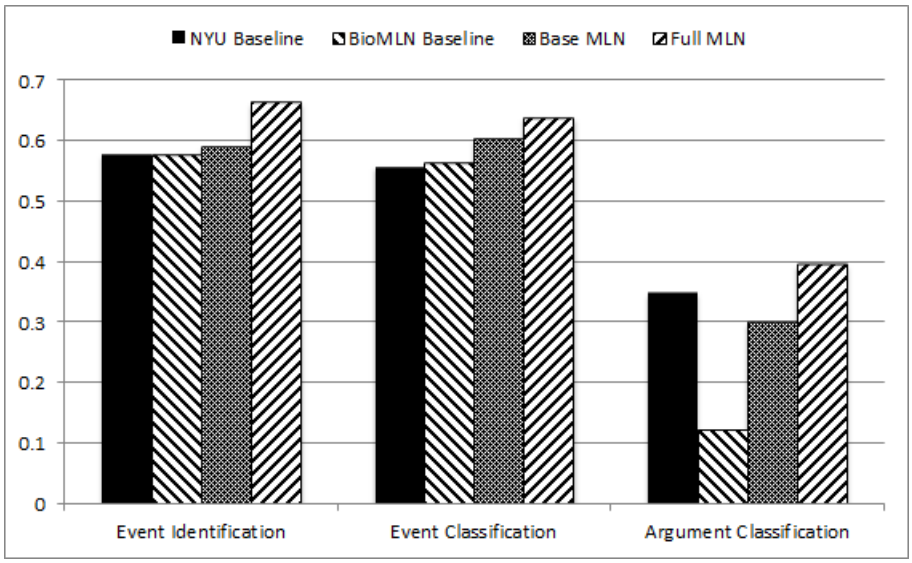


Figure 5.1: Comparison of Results in F score

also gain improvement due to the global formulas. Though there is improvement after adding event correlation information, much effort should be made to reduce the solution space when adding soft global formulas.

| | event identification | | | event classification | | | argument classification | | |
|---|---|---|---|---|---|---|---|---|---|
| | P | R | F | P | R | F | P | R | F |
| worst | 0.620 | 0.600 | 0.610 | 0.608 | 0.588 | 0.598 | 0.551 | 0.291 | 0.381 |
| optimum | 0.724 | 0.697 | 0.710 | 0.697 | 0.670 | 0.683 | 0.591 | 0.371 | 0.456 |
| average | 0.682 | 0.664 | 0.672 | 0.659 | 0.641 | 0.650 | 0.547 | 0.323 | 0.405 |

Table 5.14: Cross event within two consecutive sentences

### 5.3.6 Results of Event Classification

Table 5.15 shows the result of event classification. This result is the one with optimum performance in Table 5.14. There are 6 types of events not appearing in

this table because they are not in the testing set and our system does not misclassify them.

One of the possible reasons the performance of events like *Sue*, *Demonstrate* and *End-Org* is so high is that these triggers have more specific meanings. Taking the *Sue* event as an example, most of the *Sue* events in the ACE 05 English corpus are the words "*sue*" and "*lawsuit*". The performances of *Die*, *Attack*, *Meet* and *Transport* events are much higher than the average performance which is about 65%. These 4 types of events have more samples in the corpus. This is because they have more training samples so that the model we trained is more close to the real world. However, the performance of the *End-Position* and *Transfer-Money* events, which also have almost the same amount of samples as the *Meet* event, is much lower than the average performance. This is because the expressions of these events are very flexible. For example, we can say *He got fired yesterday* to express an *End-Position* event. However, we can also say *He got removed from the company yesterday* or *He was forced to step down from the company*. So the training samples are not enough for the various expressions.

| Event | F | K | S | C | Event | F | K | S | C |
|---|---|---|---|---|---|---|---|---|---|
| Sue | 1.000 | 6 | 6 | 6 | Phone-Write | 0.545 | 12 | 10 | 6 |
| Demonstrate | 1.000 | 2 | 2 | 2 | Appeal | 0.500 | 2 | 2 | 1 |
| End-Org | 1.000 | 1 | 1 | 1 | Transfer-Money | 0.476 | 8 | 13 | 5 |
| Die | 0.862 | 31 | 27 | 25 | Be-Born | 0.400 | 3 | 2 | 1 |
| Arrest-Jail | 0.800 | 2 | 3 | 2 | Transfer-Ownership | 0.250 | 5 | 3 | 1 |
| Attack | 0.739 | 82 | 83 | 61 | Start-Position | 0.000 | 8 | 1 | 0 |
| Meet | 0.720 | 13 | 12 | 9 | Release-Parole | 0.000 | 1 | 1 | 0 |
| Transport | 0.699 | 66 | 57 | 43 | Sentence | 0.000 | 1 | 1 | 0 |
| Declare-Bankruptcy | 0.667 | 4 | 5 | 3 | Divorce | 0.000 | 0 | 1 | 0 |
| Start-Org | 0.667 | 2 | 1 | 1 | Extradite | 0.000 | 0 | 1 | 0 |
| Injure | 0.615 | 5 | 8 | 4 | Execute | 0.000 | 0 | 1 | 0 |
| End-Position | 0.600 | 7 | 3 | 3 | Marry | 0.000 | 0 | 1 | 0 |
| Elect | 0.600 | 3 | 7 | 3 | Nominate | 0.000 | 0 | 1 | 0 |
| | | | | | Charge-Indict | 0.000 | 0 | 1 | 0 |

Table 5.15: F score of Event Classification
F=F score, K=#key samples,
S=#system samples,C=#correct samples

### 5.3.7 Results of Argument Classification

Table 5.16 shows the performance of argument classification corresponding to the optimum dataset in Table 5.14.

The performance of argument classification is closely related with the performance of its corresponding event classification. We can see from the above results that the performance of the *Victim* argument in terms of F-score is high as the performance of the *Die* event is high. Though the *Injure* event also contains the *Victim* argument, the number of *Injure* events in this testing set is small. Therefore, the influence of the *Injure* events on the performance of the *Victim* argument is low. We can also see that although the *Place* argument occupies the largest portion

of the whole argument, its performance is fairly low. One of the reasons for this maybe that the entities that play as *Place* arguments are too far away from their corresponding triggers, making it difficult for our system to recognize them.

| **Argument** | **F** | **K** | **S** | **C** | **Argument** | **F** | **K** | **S** | **C** |
|---|---|---|---|---|---|---|---|---|---|
| Defendant | 0.800 | 5 | 5 | 4 | Artifact | 0.388 | 66 | 37 | 20 |
| Position | 0.750 | 5 | 3 | 3 | Place | 0.337 | 49 | 34 | 14 |
| Victim | 0.750 | 34 | 30 | 24 | Agent | 0.296 | 21 | 6 | 4 |
| Destination | 0.736 | 47 | 40 | 32 | Target | 0.286 | 30 | 12 | 6 |
| Giver | 0.667 | 8 | 7 | 5 | Plaintiff | 0.222 | 8 | 1 | 1 |
| Recipient | 0.667 | 7 | 5 | 4 | Attacker | 0.211 | 26 | 12 | 4 |
| Org | 0.615 | 7 | 6 | 4 | Time-Holds | 0.000 | 2 | 0 | 0 |
| Origin | 0.571 | 10 | 4 | 4 | Seller | 0.000 | 2 | 0 | 0 |
| Instrument | 0.522 | 13 | 10 | 6 | Beneficiary | 0.000 | 1 | 1 | 0 |
| Crime | 0.500 | 2 | 2 | 1 | Vehicle | 0.000 | 3 | 6 | 0 |
| Money | 0.500 | 1 | 3 | 1 | Time-Starting | 0.000 | 1 | 0 | 0 |
| Adjudicator | 0.500 | 3 | 1 | 1 | Buyer | 0.000 | 5 | 0 | 0 |
| Person | 0.429 | 18 | 10 | 6 | Time-Before | 0.000 | 1 | 0 | 0 |
| Time-Within | 0.417 | 43 | 29 | 15 | Sentence | 0.000 | 1 | 1 | 0 |
| Entity | 0.410 | 50 | 33 | 17 | Time-Ending | 0.000 | 3 | 0 | 0 |
| | | | | | Time-At-Beginning | 0.000 | 2 | 0 | 0 |

Table 5.16: F score of Argument Classification
F=F score, K=#key samples,
S=#system samples,C=#correct samples

# Chapter 6

# Conclusion

This chapter concludes the thesis and presents possible research directions that future work may take.

## 6.1 Conclusion

This thesis aims at extracting a specific set of events from text documents. Normally, there are three objectives in event extraction: event identification, event classification and argument classification. This thesis has conducted a comprehensive literature review to trace the development of research work on event extraction. Moreover, we have proposed a new unified Markov logic network (MLN) inspired by the MLN for the bio-molecular event extraction task. Extensive experiments have been conducted to evaluate the performance of our framework on the ACE 05 English corpus.

The experimental results clearly show that the performance of our framework has exceeded that of state-of-the-art sentence level system. Specifically, we obtained the following conclusions:

- Our new unified MLN outperforms the BioMLN, which shows that MLNs for bio-molecular event extraction could not be directly applied to generic event extraction and the new proposed formulas can effectively improve the performance of the generic event extraction. Compared with the state-of-the-art system, the full MLN gained about 8% improvement in F-score for event identification and classification, and about 5% improvement in F-score for argument classification.

- We encode event correlation information which is helpful for generic event extraction. Experimental result shows that this information can lead to statistical significant improvement.

## 6.2 Future Work

Based on our experience with the framework mentioned in Section 5, we would like to improve our framework in MLNs in the following areas.

Exploiting a wider scope of information to help predict events is a promising direction. Yao et al. (2012) developed a topic model to disambiguate word sense ambiguity. Following this approach, we can construct a topic model to partition the potential trigger words into different sense-clusters given different contexts. Then this kind of partition information can be used as features and be incorporated into

our framework. Since the sense-clusters are generated by using document-level information, this will help to disambiguate the sense ambiguity in trigger words.

In real applications, event coreference is performed after event extraction. Since the purpose of event coreference is to predict the relationship between events, it is highly related with the event extraction task. Therefore, we can integrate this task into our framework to enhance the performance of our system. Specifically, if an event $e_1$ can be tagged with high accuracy, and another potential event $e_2$ is also presented in the document, and shares the same arguments with $e_1$, then $e_2$ can be also tagged correctly with high probability.

In the generic event extraction task, we have found that almost all the events share some common argument types, such as *Place* and *Time* (in fact, there are several kinds of *Time* arguments, but for simplicity, we refer to all of them here as *Time* arguments). Therefore, we can do some statistical analysis to find effective patterns for predicting these two kinds of arguments. In MLN, we can write some specific formulas by replacing some variables with constants. For example, we can define $dict(i, +e, prec) \wedge word(h, \text{“}in\text{”}) \wedge entity(id, h{+}1, \text{“}Place\text{”}) \Rightarrow role(i, id, \text{“}Place\text{”})$, which indicates that if a word occurring before an entity whose type is *Place*, then this word probably plays a *Place* role in a potential event.

By implementing the above approaches, the performance of our framework would be further improved. These issues will be deferred to our future work.

# Bibliography

David Ahn. 2006. The stages of event extraction. In *Proceedings of the Workshop on Annotating and Reasoning about Time and Events*, pages 1–8. Association for Computational Linguistics.

Mary Elaine Califf. 1998. *Relational learning techniques for natural language information extraction*. Ph.D. thesis, Citeseer.

H.L. Chieu and H.T. Ng. 2002. A maximum entropy approach to information extraction from semi-structured and free text. In *Proceedings of the National Conference on Artificial Intelligence*, pages 786–791. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999.

D. Ciravegna et al. 2001. Adaptive information extraction from text by rule induction and generalisation.

K. Crammer and Y. Singer. 2003. Ultraconservative online algorithms for multiclass problems. *The Journal of Machine Learning Research*, 3:951–991.

Marie-Catherine De Marneffe, Bill MacCartney, and Christopher D Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of LREC*, volume 6, pages 449–454.

2005. The ace 2005 (ace05) evaluation plan.

LI Pei Feng, ZHU Qiao Ming, DIAO Hong Jun, and ZHOU Guo Dong. 2012. Joint modeling of trigger identification and event type determination in chinese event extraction.

Aidan Finn and Nicholas Kushmerick. 2004. *Multi-level boundary classification for information extraction*. Springer.

David Fisher, Stephen Soderland, Fangfang Feng, and Wendy Lehnert. 1995. Description of the umass system as used for muc-6. In *Proceedings of the 6th conference on Message understanding*, pages 127–140. Association for Computational Linguistics.

D. Freitag and N. Kushmerick. 2000. Boosted wrapper induction. In *Proceedings Of The National Conference On Artificial Intelligence*, pages 577–583. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999.

Ralph Grishman, David Westbrook, and Adam Meyers. 2005. Nyus english ace 2005 system description. In *Proc. ACE 2005 Evaluation Workshop*. Washington.

Yu Hong, Jianfeng Zhang, Bin Ma, Jianmin Yao, Guodong Zhou, and Qiaoming Zhu. 2011. Using cross-entity inference to improve event extraction. In *Proceedings of*

*the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 1127–1136. Association for Computational Linguistics.

Ruihong Huang and Ellen Riloff. 2012. Bootstrapped training of event extraction classifiers. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 286–295. Association for Computational Linguistics.

H. Ji and R. Grishman. 2008. Refining event extraction through cross-document inference. *Proc. ACL 2008*.

Heng Ji, Ralph Grishman, Hoa Trang Dang, Kira Griffitt, and Joe Ellis. 2010. Overview of the tac 2010 knowledge base population track. In *Third Text Analysis Conference (TAC 2010)*.

S. Liao and R. Grishman. 2010a. Using document level cross-event inference to improve event extraction. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 789–797. Association for Computational Linguistics.

Shasha Liao and Ralph Grishman. 2010b. Filtered ranking for bootstrapping in event extraction. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 680–688. Association for Computational Linguistics.

Shasha Liao and Ralph Grishman. 2011a. Acquiring topic features to improve event extraction: in pre-selected and balanced collections. In *Proceedings of the Conference on Recent Advances in Natural Language Processing, Hissar, Bulgaria*.

Shasha Liao and Ralph Grishman. 2011b. Using prediction from sentential scope to build a pseudo co-testing learner for event extraction. In *Proceedings of the 5th International Joint Conference on Natural Language Processing (IJCNLP 2011)*, pages 714–722.

Wei Lu and Dan Roth. 2012. Automatic event extraction with structured preference modeling. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 835–844. Association for Computational Linguistics.

David McClosky, Mihai Surdeanu, and Christopher D Manning. 2011. Event extraction as dependency parsing for bionlp 2011. In *Proceedings of the BioNLP Shared Task 2011 Workshop*, pages 41–45. Association for Computational Linguistics.

Hoifung Poon and Lucy Vanderwende. 2010. Joint inference for knowledge extraction from biomedical literature. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 813–821. Association for Computational Linguistics.

M. Richardson and P. Domingos. 2006. Markov logic networks. *Machine learning*, 62(1):107–136.

Sebastian Riedel. 2008. Improving the accuracy and efficiency of map inference for markov logic. In *Proceedings of the 24th Annual Conference on Uncertainty in AI (UAI '08)*, pages 468–475.

Dan Roth and Wen-tau Yih. 2001. Relational learning via propositional algorithms: An information extraction case study. In *International Joint Conference on Artificial Intelligence*, volume 17, pages 1257–1263. LAWRENCE ERLBAUM ASSOCIATES LTD.

S. Soderland. 1999. Learning information extraction rules for semi-structured and free text. *Machine learning*, 34(1):233–272.

Limin Yao, Sebastian Riedel, and Andrew McCallum. 2012. Unsupervised relation discovery with sense disambiguation. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 712–720. Association for Computational Linguistics.