Relation Extraction and its Application to Question Answering

by

Ying Xu

A thesis submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

Department of Computing Science

University of Alberta

Abstract

Information extraction, extracting structured information from text, is a vital component for many natural language tasks such as question answering. It generally consists of two components: (1) named entity recognition (NER), identifying noun phrases that are names of organizations, persons, or countries; and (2) relation extraction, extracting relations between entities. In this dissertation, we assume the entities are given, and concentrate on the relation extraction task.

Traditional relation extraction task seeks to confirm a predefined set of relations in a text, such as the employment or family relation. These systems are difficult to extend by including additional relations. In contrast, the open information extraction (Open IE) task attempts to extract all relations, using words in sentences to represent the relations. My dissertation focuses on Open IE.

We first proposed a tree kernel based-Open IE system that achieved state of the art performance. One advantage of the tree kernel model is that it exploits information in syntactic parse trees without feature engineering. After observing the importance of words in relation extraction, we then incorporated word embeddings into the tree kernel and improved the system's performance.

However, previous systems have not considered implicit relations, i.e., relations implied in noun phrase structures such as *Germany's people, Google Images*, and *Shakespeare's book*. We call this type of structure *nested named entities*. To study the implicit relation phenomenon, we automatically extracted thousands of instances of training data from Wikipedia. We demonstrate the feasibility of recovering implicit relations with a supervised classification model. Our data and model provides a baseline for future work on this

task.

Last but not least, to show the effect of our relation extraction systems, we built an Open IE-based question answering system and achieved promising results. Our analysis indicates the weakness of the current Open IE systems, the role of our information extraction results, and gives directions for improvement.

We can only see a short distance ahead, but we can see plenty there that needs to be done.

– Alan Turing.

Acknowledgements

I would like to begin by thanking my supervisor Prof. Randy Goebel. I would not have made it this far without his belief in me and his guidance. He is the one who guided me to become a researcher. I would also like to thank my supervisor committee members, Prof. Greg Kondrak and Prof. Denilson Barbosa. I started the Open IE topic with Denilson. I learned so much about how to do research and how to write papers from Greg. It is thrilling to work with these people that are passionate with research. They are like my supervisors without the title.

I would also like to thank my other committee members, Dr. Christoph Ringlstetter, Prof. Dale Schuurmans, and Prof. Maosong sun, for their useful feedbacks on the dissertation.

I would also like to thank many others that I have been working with, Dr. Mi-Young Kim, Prof. Yusuke Miyao, Dr. Pascual Martínez. It's such a great opportunity to learn from so many great researchers and be friends with so many nice people.

I would also like to thank our NLP group members, Dr. Shane Bergsma, Dr. Sittichai Jiampojamarn, Dr. Mohammad Salameh, Dr. Garrett Nicolai, and Bradley Hauer, who gave me useful comments on my work from time to time.

Lastly but not least, I would like to thank my parents for making me who I am today. I would like to thank my husband, Kevin Quinn, who has cooperated with me on many projects, understands me, and encourages me to pursue my dream.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Recent research in natural language processing (NLP) has provided a glimpse into the vast potential of Artificial Intelligence. IBM's Watson, a computer system capable of answering natural language questions, defeated the two human champions on the TV quiz program Jeopardy!. An instance of its question answering ability is shown below:

> *He was a bank clerk in the Yukon before he published "Songs of a Sourdough" in 1907.*

The focus of the question is *he*. Watson is able to find the person of the description from its knowledge base.

A vital component of such question answering (QA) systems is information extraction, which extracts structured information from natural language text. The structured information usually focuses on noun phrases that the public is interested in, such as PERSON, ORGANIZATION, and LOCATION. These noun phrases are referred to as named entities. The task of recognizing those noun phrases and classifying them is called named entity recognition (NER).

The **structured information** is represented by relations between entities, relations between relations, etc. What is a relation? According to the top definition returned by Google's search engine, it is "*the way in which two or more concepts, objects, or people are connected; a thing's effect on or relevance to another.*" For information extraction tasks, consider this example: the sentence "*Barack Obama is the president of the United States*" contains the

relation *president* between the entities *Barack Obama* and *the United States*. The task of identifying relations between entities is called relation extraction. In this dissertation, my research will concentrate on relations between two entities within one sentence, even though there can be relations indicated across sentences.

Information extraction tasks are highly application or domain oriented, as such, there are no fixed definition of named entities. For example, general named entity recognition concentrates on entities such as a person's name, companies name, and countries. Normally, NER will not extract *tobacco* as a named entity, as in the example sentence "*He got high from tobacco in early days.*" However, in the clinical domain *tobacco* might be considered as an entity in sentences such as "*Tobacco use is a leading cause of cancer and of death from cancer.*" A similar problem exists in relation extraction.

Traditional relation extraction attempts to extract a fixed set of relations, such as *is-a*, *employ*, *family*, and *location*. However it is impossible to define a complete set of relations in natural language. There are occasions when we need additional relations or when we need to further divide relations. For instance, within the *family* relation we can be more specific such as using a *siblings-with* or *parents-of* relation.

Banko et al. [2007] proposed a new task, which they called open information extraction (Open IE), which aims to extract all potential relations. Unlike traditional relation extraction, it does not classify relations to a specific set such as *family*, *employ*. Instead it uses explicit words in sentences as relations. For example, in the previous sentence "*Barack Obama is the president of the United States*," traditional relation extraction will classify the relation between *Barack Obama* and *the United States* as *employ*, while Open IE will use *president* as the relation. My research began with the aim of improving existing Open IE systems.

Early Open IE systems concentrated on verb relations, but ignored noun relations [Banko et al., 2007, Fader et al., 2011], such as ***president*** in the expression "***U.S.*** *president* ***Obama***." They claimed that verb relations represent the majority of relations. This could be due to their dependence upon

noun phrase chunkers which will extract *U.S. president Obama* as a single entity. My study begins by manually annotating sentences from news text with relations between named entities, where the entities are detected by a more sophisticated NLP tool, the Stanford NER [Finkel et al., 2005]. According to my manual annotation of 750 sentences in the English Penn Treebank, 28% are implicit relations, i.e., relations without words or with prepositions. For instance, there is a *nationality* relation implied in "*Wayne Gretzky of Canada*" between the person *Wayne Gretzky* and the country *Canada*. Less than 1% of the relations are adjectives, such as the *age* relation in "*He was 6 years old.*" 71% of the relations are noun or verb phrases. Of the 71%, 60% are noun relations and 40% are verbal[1]. One example of noun relations is the *president* relation in the expression "*U.S. president Obama.*" One example of verb relations is the *graduate* relation from the sentence "*Ying graduates from University of Alberta.*" My research exploits these observations.

There are two types of relation extraction models; pattern based or machine learning based.

Pattern based relation extraction methods typically use rule matching to extract relations. For example, if in one sentence, one argument is the subject of a verb, the other argument is the object of a verb, then the verb and the two arguments form a relation triple. The current literature [Mesquita et al., 2013] largely shows that such rules or patterns are constructed by hand.

Alternatively, supervised machine learning approaches replace the manual rule or pattern construction with approaches that use selected attributes of text as inputs to learning models that help identified "average" or "typical" hypotheses on relations. In the learning process, training data can be used to assign weights to different features. Features are attributes of an instance that may have effect on the task, which can be the previous example rule. When labeling a test instance, machine learning models combine the effect of features according to the learned weights.

This dissertation concentrates on the second paradigm. Although pattern based systems can also learn patterns from training corpora, they are not as

[1]Relations that are represented by verbs or verb phrases.

flexible as supervised machine learning models. If we take conditions such as *if one entity is a subject* and *if one entity is an object* as features, pattern-based models use only several features, while supervised machine learning models can use more than thousands of features.

Our first Open IE system tackles the problem that previous Open IE systems only concentrated on verb relations and ignored others. Our system extracts both noun and verb relations. We extract features from dependency paths and adapt an SVM dependency tree kernel model [Moschitti, 2006] as the basis for our system. Selecting relevant features from a parse tree for semantic tasks is difficult. SVM tree kernels avoid extracting explicit features from parse trees by calculating the inner product of the two trees. Tree kernels have been used in traditional RE and have helped achieve state of the art performance. However, one challenge of using tree kernels on Open IE is that the lexicon of relations is much larger than those of traditional RE, making it difficult to include the lexical information as features. As a first step, we propose an unlexicalized tree structure for Open IE. As far as we know, this is the first time an SVM tree kernel has been applied in Open IE. Experimental results on multiple datasets show that this system outperforms previous state-of-the-art systems: REVERB [Fader et al., 2011] and OLLIE [Mausam et al., 2012].

Our second Open IE system embeds lexicalized information into a tree kernel. One reason that previous Open IE systems did not include lexical information as features is because it creates a sparsity issue. The training set of Open IE systems is not large, e.g., several thousands. If we include lexical features, a significant amount of words will be absent in the training data or will be infrequent and their corresponding parameters, i.e., effect on the task output, will be poorly estimated. However, the error analysis on previous work suggests that Open IE would benefit from lexical information because the same syntactic structure may correspond to different relations. For instance, the relation <*Annacone, coach of, Federer*> is correct for the sentence "*Federer hired Annacone as a coach,*" but not for the sentence "*Federer considered Annacone as a coach,*" even though they have the same dependency path

structure [Mausam et al., 2012]. Instead of representing word features as binary values, i.e., weather the word occurs or not, we use a type of smoothed word representations: word embedding. Word embedding represents words as numerical vectors, which decrease the feature dimension dramatically, from the size of the vocabulary to the length of the word embedding.

We then address the issue of implicit relations, which are implied in the structure of the sentence. For example, the noun phrase *University of Alberta* implies a relation *locate* between the two entities *University of Alberta* and *Alberta*; *Google Images* implies *Google* owns the product *Google Images*. These relations are ignored in the current relation extraction literature. The relations are implied in nested named entities, a structure where one entity is resides within another. Current named entity recognition systems consider named entities in a flat structure with no overlap. As the relation extraction task is dependent on the entity extraction task, implicit relations are largely ignored. However, as shown by our annotated data mentioned previously, 28% of the relations are implicit. We believe that learning implicit relations in nested named entities can provide insight into implicit relation extraction in other noun phrase structures such as "*Wayne Gretzky of Canada*." Their extraction will complete the relation extraction task and benefit tasks such as question answering and textual entailment.

The final component of our work is in applying our Open IE systems to the question answering task. There are two critical components in our question answering system. One is the construction of a knowledge base. Here a knowledge base means a database with relation triples <E1, R, E2>, where E1 and E2 are two entity arguments of the relation R. The other is paraphrase, which creates a mapping from the wording of questions to that of the knowledge base. For example, if the knowledge base is Freebase, then we should map the relation *play* in "*What character does Natalie Portman play in Star Wars?*" to the Freebase relation */film/actor/film /film/performance/film*; if the knowledge base consists of triples from a Open IE system, we need to know that *play* can also be represented as *star as*. Our question answering system shows better performance than the previous Open IE based question

answering system [Fader et al., 2014].

In summary, there are three main contributions of this dissertation.

- First, we improved Open IE performance with SVM tree kernel models. The lexicalized tree kernel model is able to leverage both syntactic and lexical features.

- Second, we proposed a new task, implicit relation extraction, to the literature of information extraction, which has potential to improve natural language understanding. We created training data and supervised models for the task, which can be the baseline for future work on this task.

- Third, we built a question answering system based on our Open IE results. We showed that better Open IE performance leads to better question answering performance.

The rest of this dissertation is organized as follows: first I will provide the background of the information extraction and machine learning models in Chapter 2. Chapter 3 will describe our Open IE system based on an unlexicalized tree kernel. Chapter 4 presents its improved version based on a lexicalized tree kernel. Chapter 5 introduces our study on implicit relation extraction. Chapter 6 describes our question answering system. I will conclude the dissertation and propose future work for Open IE in Chapter 7.

Chapter 2

Background and Related Work

Advances in computer storage techniques have led to an explosion of digital data, a large portion of which is text. Information extraction (IE) is the task of extracting structured data from the unstructured information embedded in text. It is a preliminary step for other tasks such as question answering and user behaviour prediction. To illustrate the process of IE, I will use the following paragraph as an example (from the Wikipedia page of the 2008 Summer Olympics).

> *The 2008 Summer Olympic Games, commonly known as Beijing 2008, was a major international multi-sport event that took place in Beijing, China, from 8 to 24 August 2008. The theme song of the 2008 Olympic Games was "You and Me," which was composed by Chen Qigang, the musical director of the opening ceremony. Qiqang was born in China and has lived in France since 1984.*

The first step of IE is named entity recoginition (NER), detecting and/or classifying all the proper names mentioned in a text. What counts as a named entity depends on the domain of applications. There are three commonly used NER sets [Finkel et al., 2005]:

- 3 classes: Location, Person, Organization.
- 4 classes: Location, Person, Organization, Miscellaneous (Misc.).
- 7 classes: Location, Person, Organization, Money, Percent, Date, Time.

If using NER 4 classes set, in the previous example paragraph, *2008 Summer Olympic Games* will be tagged as MISC. *"You and Me"* will also be tagged as MISC. *Beijing* and *China* are examples of LOCATION. *Chen Qiqang* is an example of PERSON. The following are two sentences with all the entities marked according to the 4 class label set.

> *The theme song of the* [$_{MISC}$ *2008 Olympic Games*] *was* [$_{MISC}$ "*You and Me,*"] *which was composed by* [$_{PER}$ *Chen Qigang*], *the musical director of the opening ceremony.* [$_{PER}$ *Qiqang*] *was born in* [$_{LOC}$ *China*] *and has lived in* [$_{LOC}$ *France*] *since 1984.*

The second step of IE is relation extraction (RE): finding (and classifying) relations between the entities identified in text. Traditional relation extraction systems first define a fixed set of relations, and then focus on identifying relations that belong to the set while ignoring others. One popular set of relations is from the ACE (Automatic Context Extraction) standard [Doddington et al., 2004], which is developed by NIST in 1999. It has relations such as *Affilation, Geospatial*, and *Part-of*. In the previous example, there are relation triples such as <*Qiang, Gen-Affiliation, China* > and <*Qiang, Gen-Affiliation, France*> that belong to the set.

Notice that there are relations that are not tagged with the traditional relation extraction standard. For example, <*"You and Me", theme song, 2008 Olympic Games*> and <*"You and Me", compose, Chen Qigang*> will not be extracted by any of the traditional relation extraction systems because they do not belong to any category defined in the ACE standard. To tackle this problem, Banko et al. [2007] proposed a new task, open information extraction (Open IE), which attempts to extract all relations in text.

My dissertation concentrates on Open IE. In the following section, I will introduce related work on traditional relation extraction and Open IE. As both of my Open IE systems use SVM tree kernel models, I will first introduce tree kernel models.

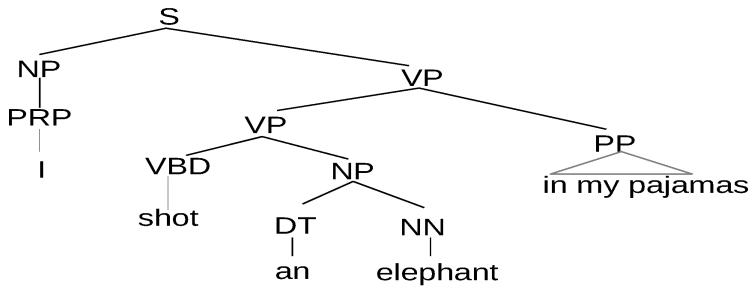


Figure 2.1: A constituent parse tree example.

2.1 Tree Kernels

Most IE systems incorporate machine learning models to learn to tag relations in sentences. The quality of such models depends heavily on the choice of text features. There are usually two levels of features for IE tasks. The first level includes lexical features. Examples are words and word shapes, e.g., whether a word is Capitalized. The second level includes syntactic features. Examples are part-of-speech (POS) tags and syntactic parse trees. POS tags classify words into categories such as verbs (VB), nouns (NN), and adjectives (JJ). Syntactic parsing assigns a syntactic structure, a tree, to a sentence [Jurafsky and Martin, 2000].

There are two popular types of parsing in NLP. One is constituent parsing, the other is dependency parsing. Constituent parsing derives a constituency structure from a sentence. The idea of constituency is that groups of words may behave as a single unit or phrase, called a constituent [Jurafsky and Martin, 2000]. Figure 2.1 is an example of constituent parse trees. In the figure, *NP* means *noun phrase*, *VP* means *verb phrase*, and *PP* means *preposition phrase*. The detailed structure of "*in my pajamas*" is hidden.

Another popular parsing type is dependency parsing. Instead of grouping words into constituents, dependency parsing represents the sentence structure by adding links between words. Figure 2.2 shows an example of a dependency structure. In the example, there is a dependency relation between *shot* and *I*: *shot* is the dominant word, *I* is the dependent word, and the relation is *nsubj*, which is an abbreviation for *noun subject*.

While it is relatively easy to identify relevant lexical and POS features for

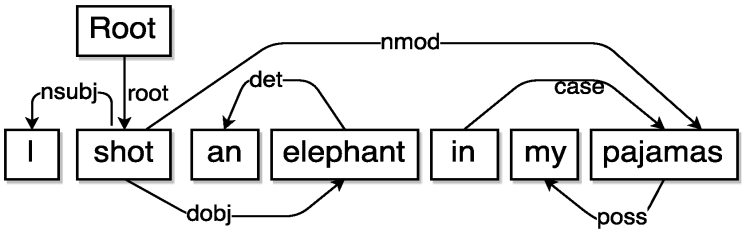


Figure 2.2: A dependency parse tree example.

the IE task, it is difficult to select relevant features from a parse tree. Assume we extract all possible subtrees in a tree as features, if the subtree size is d, then a tree T can be represented by a vector $\mathbf{h}(T) = \{h_1(T), h_2(T), ...h_d(T)\}$, where $h_i(T)$ is the number of times subtree i occurs in the tree. Extracting all the subtrees is exponential in the size of the tree nodes.

The use of tree kernels avoids extracting explicit features from parse trees by calculating the inner products of two trees directly. They can be used in machine learning models which classify instances by calculating the similarity of pair instances, i.e., inner products. Examples of these models are perceptron, support vector machine (SVM), and principal component analysis (PCA). A tree kernel function over two trees T_1 and T_2 is:

$$K(T_1, T_2) = \sum_{n1 \in N_{T_1}} \sum_{n2 \in N_{T_2}} \Delta(n_1, n_2) \tag{2.1}$$

where N_{T_1} and N_{T_2} are the set of trees' nodes [Collins and Duffy, 2001]. The Δ function is computed recursively. It provides the basis for identifying subtrees of nodes, which is the essential distinction between different tree kernel functions.

Two popular tree kernels are the Sub Tree (ST) and the SubSet Tree (SST) kernels. For the ST kernel, a subtree includes any node of a tree along with all its descendants. Figure 2.3 shows the ST set for the node *VP*. SSTs [Collins and Duffy, 2002] are more general, in which not all the descendants are necessarily included; the only constraint is that SSTs must be generated by the same grammar rules which generate the original tree, which means the entire

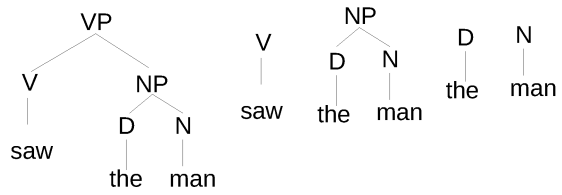


Figure 2.3: An example of a tree set for the Sub Tree kernel (ST).

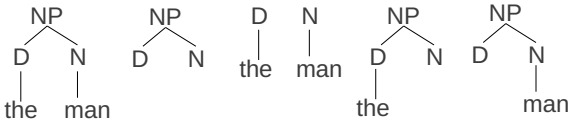


Figure 2.4: An example of a tree set for the SubSet Tree kernel (SST).

child set for a node must be included. Figure 2.4 shows the SST set for the node *NP*. Following is the Δ function of the SST kernel.

1. If the production (context free grammar) of n_1 and n_2 are different, then $\Delta(n_1, n_2) = 0$,

2. else if n_1 and n_2 are pre-terminals (POS) and they are the same, then $\Delta(n_1, n_2) = 1$,

3. else the value is calculated recursively down the tree:

   $\Delta(n_1, n_2) = \prod_{j \in nc(n_1)}(1 + \Delta(ch(n_1, j), ch(n_2, j))$,

where nc is the number of children of a node, and $ch(n_1, j)$ is the jth child.

To illustrate the difference between SST and ST, consider the $\Delta(VP_1, VP_2)$ values of Tree 1 and Tree 2, in Figure 2.6. The value of ST will be 0 because the trees below the VP nodes are not the same. The value of SST will be $[1 + \Delta(V_1, V_2)][1 + \Delta(NP_1, NP_2)]$, where V_1 means the subtree with root V in Tree 1. $\Delta(V_1, V_2) = 1$, $\Delta(NP_1, NP_2) = 0$, which leads to $\Delta(VP_1, VP_2) = 2 * 1 = 2$.

One problem of the SST kernel is that, although it is suitable for constituent parse trees, it performs poorly when applied to dependency parse trees for

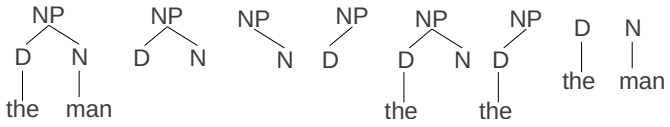
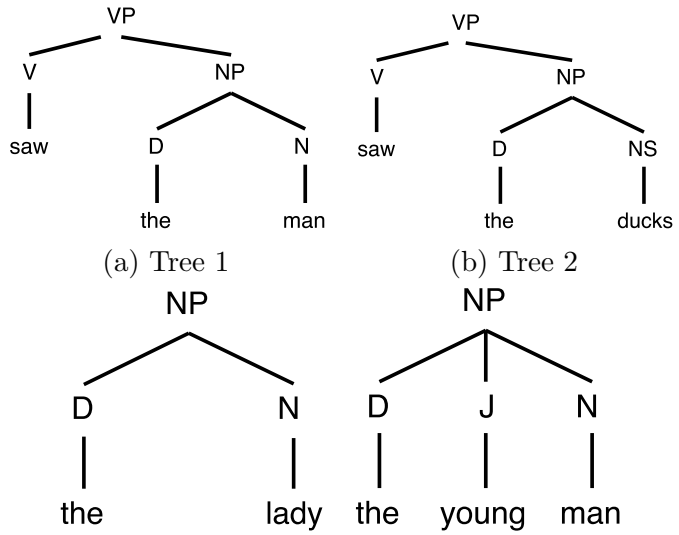


Figure 2.5: An example of a tree set for the Partial Tree kernel (PTK).



(a) Tree 1 (b) Tree 2

(c) Tree 3 (d) Tree 4

Figure 2.6: Example trees for ST, SST, and PTK value calculation.

semantic tasks [Moschitti, 2006]. Moschitti [2006] proposed a more general tree kernel, the partial tree kernel (PTK), which can be used with both constituent and dependency parse trees. It generalizes the constraint that the grammar set for generating the subtrees should be the same as the one generating the original tree. When the node labels of n_1 and n_2 are the same, the Δ function of PTK is:

$$\Delta(n_1, n_2) = 1 + \sum_{J_1, J_2, l(J_1)=l(J_2)} \prod_{i=1}^{l(J_1)} \Delta(c_{n_1}(J_{1i}), c_{n_2}(J_{2i}))) \tag{2.2}$$

$\Delta = 0$ when the node labels are different. c_{n_1} and c_{n_2} are child sequences of nodes n_1 and n_2 respectively. $J_1 =< J_{11}, J_{12}, J_{13}... >$ and $J_2 =< J_{21}, J_{22}, J_{23}... >$ are index sequences of the two child sequences, J_{1i} and J_{2i} are the i-th children of the two sequences. For example, if a tree is (S(DT JJ N)) then J for the node S is chosen from [(DT), (JJ), (N), (DT JJ), (DT N), (JJ N)...]. $l()$ denotes the sequence length, so in the previous example, if $J_1 = (DTN)$, i.e., $(1, 3)$, then $l(J_1) = 2$.

To illustrate the difference between SST and PTK, consider the $\Delta(NP_3, NP_4)$ of Tree 3 and Tree 4, in Figure 2.6. The value of SST will be 0 because the production of NP_3 is (D, N), while the production of NP_4 is (D, J, N). For PTK,

there are three sequence pairs that have non zero value: (D_3, D_4), (N_3, N_4), and $((D_3\ N_3), (D_4\ N_4))$. The value of PTK is:

$$\Delta(NP_3, NP_4) = 1 + [\Delta(D_3, D_4) + \Delta(N_3, N_4) + \Delta(D_3, D_4) \times \Delta(N_3, N_4)]. \quad (2.3)$$

As $\Delta(D_3, D_4) = 1$ and $\Delta(N_3, N_4) = 1$, the value of $\Delta(NP_3, NP_4) = 4$.

Moschitti [2006] also added two decay factors μ and λ, both of which are in the range of (0, 1). μ is for the height of the tree, penalizing large trees. λ is for the length of the child sequences, penalizing sequences with gaps. The modified function is:

$$\mu(\lambda^2 + \sum_{J_1, J_2, l(J_1)=l(J_2)} \lambda^{d(J_1)+d(J_2)} \prod_{i=1}^{l(J_1)} \Delta(c_{n_1}(J_{1i}), c_{n_2}(J_{2i}))) \quad (2.4)$$

where d is the distance between the last child and the first child in a sequence J, $d(J_1) = J_{1l(J_1)} - J_{11}$ and $d(J_2) = J_{2l(J_2)} - J_{21}$. Consider $J_1 = (DTN)$ in the previous example again, $d(J_1) = J_{12} - J_{11} = 3 - 1 = 2$.

In the previous kernels, lexical information is used as *one-hot* representation; the value is 1 when two words are the same, 0 when different. However, there are words that are more similar than others, e.g., synonyms. Intuitively, "*Jordan is mom of Kit.*" should be more similar to "*Jordan is mother of Kit.*" than to "*Jordan is dad of Kit.*" Previous kernels failed to detect this difference. Croce et al. [2011] proposed a new tree kernel, smoothing partial tree kernel (SPTK), to leverage lexical information. It is very similar to PTK, except that if n_1 and n_2 are two leaves, i.e., words, $\Delta(n_1, n_2) = \mu * \lambda * \sigma(n_1, n2)$. $\sigma(n_1, n2)$ is the function for word similarity. They represented words as word vectors created by Singular Value Decomposition [Golub and Kahan., 1965] from a word co-occurrence matrix. Word vectors are compressed representation of words. Instead of being a vector of $\{0, 1\}$, where the length of the vector is the size of the vocabulary, the word are represented as a numerical vector, such as [0.1, 0.2, 0.1, 0,...], where the length is much shorter than the size of the vocabulary and is a parameter of the model.The word similarity becomes the similarity of two word vectors.

Our proposed lexicalized tree kernel uses the popular word embedding representation for lexical information.

Despite their performance, one problem of tree kernel models is that they are slower than one-hot representation approaches. The complexity of ST and SST kernels is $O(|N_{T_1}| \times |N_{T_2}|)$ [Collins and Duffy, 2002]. The complexity of PTK is $O(p \times \rho^2 \times |N_{T_1}| \times |N_{T_2}|)$, where p is the number of different sequences and ρ is the maximum branching factor in trees T_1 and T_2 as proved in [Moschitti, 2006].This complexity causes problem when training with large dataset, e.g., more than ten thousands. With large training data and experts who can choose relevant features, one-hot representation might be a better choice.

2.2 Traditional Relation Extraction

Relation extraction (RE) is a task of discovering various semantic relations, e.g., <*Obama, president, the United States*>, from natural language text. Traditional RE is a task of extracting a specific set of relations between named entities. The Automatic Content Extraction (ACE) program's Relation Detection and Characterization (RDC) is such a task [Consortium, 2008]. It is designed to extract relations such as {*family, employee, location*,...}. For building traditional relation extraction models, every relation has manually annotated example instances. For example, "*a military base in Germany*" contains a *FAC-GPE* relation; in "*He was campaigning in his home state of Tennessee.*" there is a *PER-GPE* relation between *he* and *his home state of Tennessee*.

Many traditional RE systems use SVM tree kernel models. Besides the selection of alternative SVM tree kernels, another essential for designing traditional RE systems is to decide which portion of the sentence tree embeds the desired syntactic relational information between two named entities. For example, in Zhang et al. [2006], five approaches were studied; here are three of them.

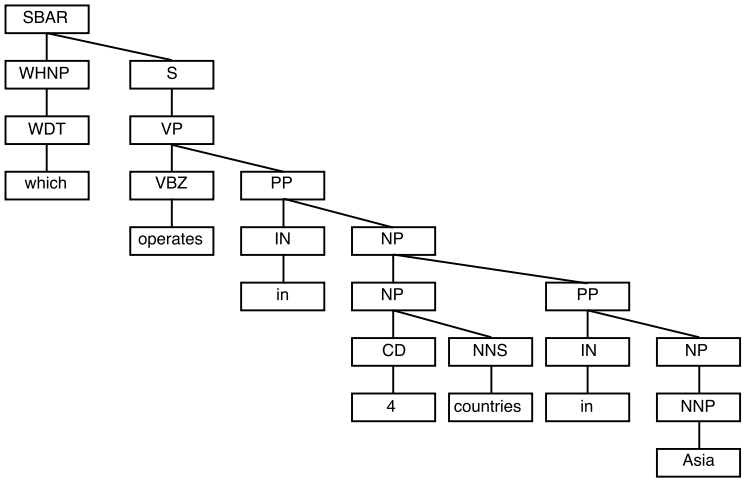
1. Minimum Complete Tree (MCT): the complete sub-tree rooted by the nearest common ancestor of the two entities under consideration.

2. Path-enclosed Tree(PET): the smallest common sub-tree including the two entities, i.e., the sub-tree which is enclosed by the shortest path between the two entities in the parse tree. These trees are sub-trees of the minimum complete trees.

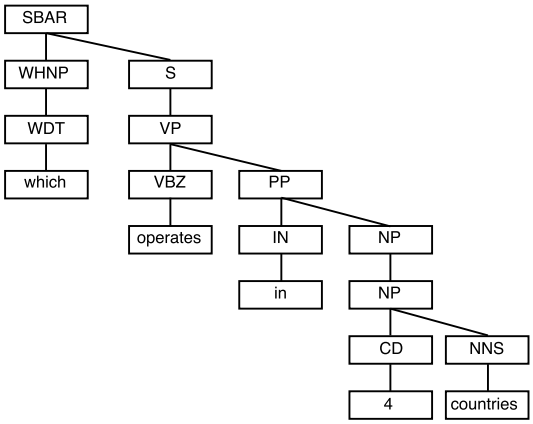
3. Context-Sensitive Path Tree(CSPT): the second one extended with the first left word of entity 1 and the first right word of entity 2. Here entity 1 is the left entity and entity 2 is the right entity.

Figure 2.7 shows the corresponding example tree fragments for the three approaches. The original sentence is "... *said Sotheby, which operates in 4 countries in Asia.*" The two entities are *which* and *4 countries.*The minimum complete tree (Figure 2.7a) starts from the nearest common ancestor of the two entities, *SBAR*. The path-enclosed tree (Figure 2.7b) trims off the preposition portion from the MCT. The context sensitive path tree (Figure 2.7c) adds fragments to include the left word *Sotheby* and the right word *in*. The experiment results on ACE 2003 and 2004 corpora showed that the path-enclosed tree was the best.

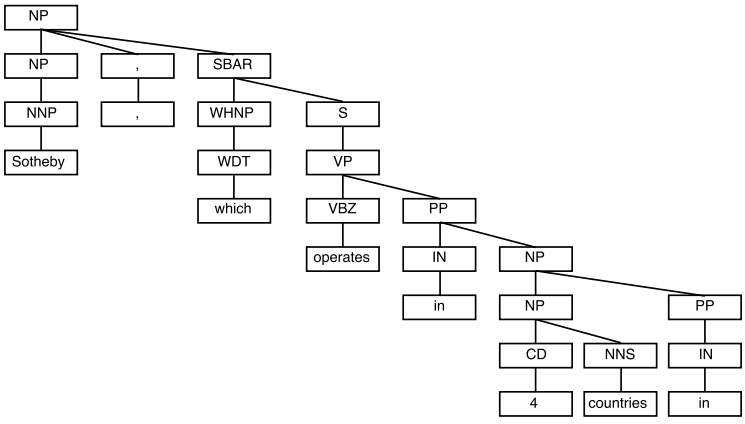
One problem of the path-enclosed tree approach is that for sentences such as "*John and Mary got married*," the path contains insufficiant information (Tree in Figure 2.8e). Zhou et al. [2007] proposed to categorize tree structures into 5 categories: embedded, PP-liked, semi-structured, descriptive, and predicate-linked structure (with examples in Figure 2.8). The first four are easily classified according to the patterns, the rest were classified into the predicate-linked category. They then used the path-enclosed tree approach for the first four categories and a specific approach, context-sensitive path (different from the previous one), to extract relevant sub-trees for the predicate-linked category. To find the context-sensitive path, they moved up along the shortest path until one predicate-head node was found, then moved down along the predicate-head path to the predicate terminals. In the example (Figure 2.8e), they moved up from *NP* to *S*, which has a verb as the head. Then include the $S \rightarrow VP \rightarrow VP$ fragment, which contains the verb *married*, into the tree. Both Zhang et al. [2006] and Zhou et al. [2007] employed the SST kernel.



(a) Minimum Complete Tree.

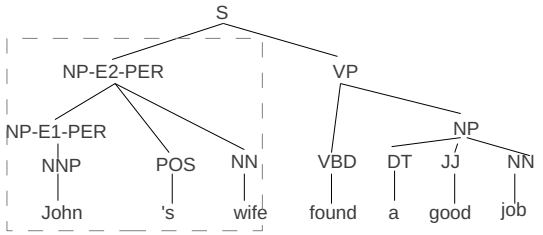


(b) Path-enclosed Tree.

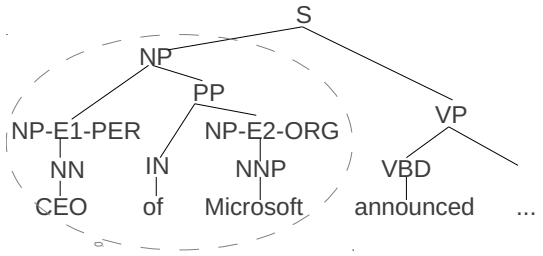


(c) Context-Sensitive Path Tree.

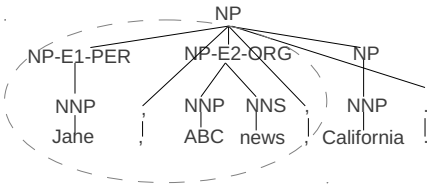
Figure 2.7: Three approaches to choose a segment of sentence trees for the traditional relation extraction in Zhang et al. [2006]. The original sentence is "... said Sotheby, which operates in 4 countries in Asia."



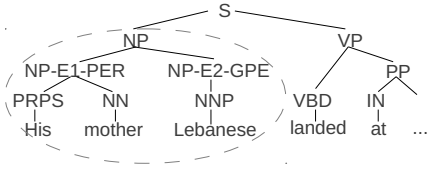
(a) embeded structure.



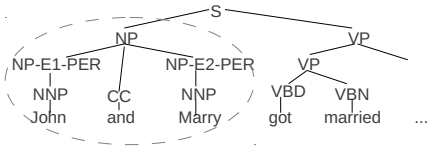
(b) PP-linked.



(c) semi-structured



(d) descriptive



(e) predicate-linked

Figure 2.8: The five tree structure categories in Zhou et al. [2007].

There are also specific strategies for choosing sub-tree structures for dependency parse trees. Bunescu and Mooney [2005] proposed a simple kernel for dependency trees. First, the shortest dependency path between two named entities were extracted as the necessary structure. They then proposed a new tree kernel. If two paths have different numbers of children, then the kernel value is 0. Otherwise, it is the kernel value sum of every child. The system abandoned structures with different path lengths, which might be the reason that the performance was worse than those of using constituent tree structures.

Nguyen et al. [2009] integrated both dependency and constituency tree structures. The partial tree kernel was employed. The combined kernel achieved the state-of-the-art performance.

Besides leveraging tree kernels, traditional relation extraction systems have employed various other techniques and features. For example, Zhou et al. [2005] explored different features such as context words, entity words, entity types, head words of entities, head words of context after chunking, the path of the parse tree that connects the two entities, and dependent words of the two entities in the dependency tree.

There are systems that use pattern extraction approaches for relation extraction. One example is the work of Xu et al. [2007]. Given seeds of a relation, i.e., relation instances, the system first extracted sentences that contained the relation arguments and assumed that the sentences contain another instances of the relation. Then patterns for the relation were learned based on the POS tags, NER tags, and dependency subtrees between the arguments on the new instances. The learned patterns are used to extract new relation instances. Their system is able to handle k-ary relations. One example pattern for the relation (prize, year, recipient) is that "the verb is *win*, the subject is a Person, the object has the head word *prize* and a year number."

Chan and Roth [2011] pointed out that identifying the syntactic relations between two entities will help classify the semantic relations. They proposed a two-step classification process for traditional relation extraction. First, relations between entity pairs were classified into five syntactic types:

1. **Premodifier** type: specify the relations between a proper adjective or a proper noun premodifier and the noun it modifies, e.g., *[the [Seattle] zoo]*

2. **Possessive** type: one entity is in a possessive case, e.g., *[[California] 's Governor]*

3. **Preposition** type: two entities are related via a preposition, e.g., *[students] in [UofA]*

4. **Formulaic** type: relations are implied in structures, for example, *[Edmonton], [Alberta]*

The classification is based on several POS patterns extracted manually according to training instance observations.

After classifying instances into the patterns, they trained two relation classification models, one on the whole training set, RE_{base}, the other trained on the instances which contain any of the first four syntactic types, RE_s. On testing, if an instance belonged to one of the four types, RE_s was used; otherwise RE_{base} was used. The results showed that using different models for different syntactic structures was better than using only one model.

2.3 Open Information Extraction

In contrast to traditional relation extraction, which specifies a set of relations to extract, open information extraction (Open IE) tries to extract all possible relations from the text[1]. The common definition of the Open IE task is a function from a sentence, s, to a set of triples, $\{< E1, R, E2 >\}$, where $E1$ and $E2$ are entities (noun phrases) and R is a textual fragment indicating a semantic relation between the two entities. Note that in this dissertation we concentrate on binary relations while there are research on n-ary relations (n>=2).

[1]We believe the definition of relation depends on the application domain of the Open IE systems.

Current approaches for Open IE fall into two categories: supervised models and pattern-based models. Supervised models extract lexical, syntactic, or semantic features and train a supervised model, such as conditional random fields(CRF) and SVM, for the task. Pattern-based models extract relations based on dependency path patterns, such as "the link between one entity and a relation word should be *nsubj*." Patterns can be automatically extracted from large training data, or designed manually. After pattern extraction, relation triples that match a certain pattern will be selected as the correct relations.

Banko et al. [2007] were the first to propose the idea of Open IE. Their system TEXTRUNNER is a self-supervised learner. The system extracts entities with a noun phrase chunker. A noun phrase chunker is a system that tags non-overlapping noun phrases in a sentence. For example, there are two noun phrase chunks, *the morning flight* and *Denver* in the sentence "*The morning flight from Denver has arrived*" [Jurafsky and Martin, 2000]. Words between two entities are extracted as candidate relations. Then entity pairs with the candidate relations are automatically tagged as positive or negative by heuristics on dependency parse information, such as "there exists a dependency chain between two entities that is no longer than a certain length." A model is trained on these instances with features such as POS tag sequences of relation words, the number of tokens in the relation, whether or not an entity is a proper noun, etc. After extracting all relation triples based on this model, a probability is assigned to every triple based on its frequency on the whole corpus. One problem with this self-supervised method is that the heuristics for training data extraction create errors, which are amplified in later stages. Banko et al. [2007] did not analyze the performance of the automatic training annotations, and their system can only extract verb relations between two entities.

Wu and Weld [2010] proposed another Open IE system, WOE, which utilized Wikipidia Infobox. To create training data, they extracted relation triples in Wikipedia pages that were also in the corresponding Infobox. These were considered as positive training instances. Then the system extracted dependency path patterns of relations from the large training data, and used these

patterns to extract new relations. The patterns included POS tags and dependency link labels. Consider the sentence "*Dan was not born in Berkeley,*" for the two entities, "*Dan*" and "*Berkeley*", the shortest dependency path was extracted to represent the candidate relation between them, *Dan* $\overset{nsubjpass}{\leftarrow}$ *born* $\overset{prep\_in}{\rightarrow}$ *Berkeley*. All adverbial, adjectival modifiers, and dependent words for the relation such as "*neg*" were appended to the path. In the previous example, the word "*not*" and "*was*" would be appended. Then lexicon words were replaced by generalized POS tags such as "N" and "V". The previous example would be generalized to "N $\overset{nsubjpass}{\leftarrow}$ V $\overset{prep}{\rightarrow}$ N." Patterns were assigned a probability according to their frequencies in the training data. This simplicity of patterns limits the system's performance. We suspect that more sophisticated supervised models will improve the performance. Note that their system can only extract verbal relations matching patterns of "subject, relation, object", although relation words can be before or after the entities.

Fader et al. [2011] have developed REVERB, a supervised model that solves the problem of incoherent or uninformative extractions of the two previous systems, TEXTRUNNER and WOE. Incoherent extractions represent relation candidates that are not meaningful. For example, TEXTRUNNER will extract "*contains omits*" as a relation in the sentence "*The guide contains dead links and omits sites.*" Uninformative extractions are cases where relations omit critical information. For example, it is not meaningful to extract the triple <Faust, made, a deal> from the sentence "*Faust made a deal with the devil.*"

Instead of extracting entities first, REVERB extracted verbal relation sequences based on a set of regular expressions. Then entities were identified around the relation sequence. After the relation triple candidate extraction, REVERB trained a model to classify the candidates as correct or not. One problem of their system is that it can only extract relation tokens between two entities. Relations such as <Nekoosa, counterbidder, International Paper> in "*Among companies mentioned by analysts as possible counterbidders for Nekoosa are International paper, Canadian Pacific Ltd. and MacMillan Ltd.*" will be ignored.

Mausam et al. [2012] are the first to generalize the constraints that relations

have to be verb phrases. They proposed an Open IE system called OLLIE. The training data was created in a bootstrapping way. They started with over 110,000 seed triples which were extracted by REVERB with high confidence. Then sentences were retrieved from webpages that contain all content words in any of the extracted triples. 18 million sentences were retrieved. They assumed that these triples in the sentences were positive instances. Then OLLIE learned relation patterns composed of dependency path and lexicon information. Relations matching the patterns were extracted. One difference of the patterns in Mausam et al. [2012] and those of Wu and Weld [2010] is that, these patterns include lexical information if they do not hold three checks such as "the relation node is between two entities." But again, they used patterns instead of supervised machine learning models.

Christensen et al. [2011] proposed to extract relations based on semantic role labeling (SRL). Semantic role labelling is a task that identifies and classifies arguments, such as subjects, objects, and time-constraints, for predicates, mainly verbs. The SRL tool they adapted was based on syntactic parsing information. The performance of the system is much better than the baseline, TEXTRUNNER, but in the sacrifice of speed. The main problem of this approach is that it can only extract verbal relations.

Akbik [2009] is another example of using dependency path information for Open IE. They brainstormed 46 general dependency path patterns from an annotated corpus of 10,000 sentences, and used these patterns to extract relations. Their patterns exclude POS information and preserve only dependency link labels. Based on a set of 4000 manually annotated wikipedia sentences, which contains 1278 relations, the system's recall is 16% and the precision is 82%. They did not compare their system to other IE systems.

Mesquita et al. [2011] was our first attempt of using Open IE model for the slot filling task in Text Analysis Conference (TAC). The task states that given around 22 relations and their training instances, extract argument 2 of a relation given argument 1. The answers are in a large set of webpages. The results showed that our Open IE systems at the moment was not as good as using traditional relation extraction for the task.

Open IE systems	pattern or ML	parsing-based	verb only
TEXTRUNNER [Banko et al., 2007]	ML	no	yes
REVERB [Fader et al., 2011]	ML	no	yes
OLLIE [Mausam et al., 2012]	pattern	yes	no
[Mesquita et al., 2013]	pattern	yes	no
[Xu et al., 2013]	ML	yes	no
[Xu et al., 2015]	ML	yes	no

Table 2.1: Summary for different Open IE systems.

Mesquita et al. [2013] proposed another dependency pattern based model for Open IE. Their model is solely based on less than 10 dependency patterns. The experiment showed that their performance was competitive and the system was quicker than other models.

Based on these studies, we have pursued several open problems in Open IE. First, sophisticated supervised machine learning models have not been explored in Open IE, which have improved performance in other NLP tasks. We adapt SVM tree kernels for Open IE, leveraging both parsing and POS tagging information. Second, how to leverage information from words, POS tags, and parsing has not been studied. We then included lexical information by proposing a lexicalized SVM tree kernel function. Third, relation forms have been restricted to explicitly mentioned words, and no implicit relation has been considered. We created a dataset with Wikipedia and Freebase, and trained a model for implicit relation extraction.

Table 2.1 lists the current Open IE systems and their characteristics: whether they are supervised machine learning based or pattern based; whether they depend on parsing; whether they extract relations other than verbs.

2.4 Evaluation Metrics

For both traditional relation extraction and open information extraction evaluation, researchers will provide datasets in which entities and their relations are annotated. The annotated relation set is called a gold standard set. The most common metrics for evaluation of relation extraction is precision, recall, and F-score.

Precision is the percentage of relations a model extracted that are correct, i.e., matches with any of the gold standard relations. Recall is the percentage of

gold standard relations that are extracted by the model. F-score is a harmonic mean of precision and recall (Equation 2.5).

$$F = 2 * \frac{precision * recall}{precision + recall} \tag{2.5}$$

However, the main problem in Open IE evaluation is the unclear definition of relations. Different datasets lead to different ranking of systems' performance. One observation from our research is that the distribution of noun and verb relations will also affect a system's performance. For example, our systems are better at noun relation extraction than OLLIE, but may be comparable in terms of verb relations extraction. Instead of creating one benchmark for all systems, we suggest using several test datasets from different annotation groups to test and compare the Open IE systems.

Chapter 3

Open Information Extraction with Tree Kernels

Proposed by Banko et al. [2007], open information extraction (Open IE) aims at extracting all general relations from text, which is different from tradition relation extraction, that aims at extracting a specific set of relations. As mentioned in Section 2.3, most early Open IE systems have targeted verbal relations, i.e., relations that are represented by verbs or verb phrases, claiming that these are the majority. However, Chan and Roth [2011] showed that only 20% of relations in the ACE program's Relation Detection and Characterization (RDC) are verbal. Our manually extracted relation triple set from the Penn Treebank shows that there are more nominal relations than verbal ones, with the ratio as 3 to 2.

This difference arises because of the ambiguity of what constitutes a relation in Open IE. It is often difficult for annotators to agree on what constitutes a relation, and which words in the sentence establish a relation between a pair of entities. For example, in the sentence "*Olivetti broke Cocom rules,*" is there a relation between *Olivetti* and *Cocom*? This ambiguity in the problem definition leads to significant challenges and confusion when evaluating and comparing the performance of different methods and systems. An example are the results in Fader et al. [2011] and Mausam et al. [2012]. In Fader et al. [2011], REVERB is reported as superior to WOE^{parse}, a system proposed in [Wu and Weld, 2010]; while in Mausam et al. [2012], it is reported the opposite.

To better answer the question, *what counts as a relation?* We designed

experiments for two tasks. The first task seeks to determine whether there is a relation between two entities (called "Binary task"). The other is to confirm whether the relation words extracted for the two entities are appropriate (the "Triple task"). The Binary task does not restrict relation word forms, whether they are mediated by nouns, verbs, prepositions, or even implicit relations. The Triple task requires an abstract representation of relation word forms. Currently we assume that relation words are nouns or verbs; in our data, these two types comprise 71% of relations. Note that the definition of entities is also ambiguous. Consider again the sentence "*Olivetti broke Cocom rules,*" both *Cocom* and *Cocom rules* can be as an entity. In this chapter, we use the Stanford NER tool to identify entities.

Inspired by the success of traditional relation extraction models that use SVM tree kernels [Bunescu and Mooney, 2005, Nguyen et al., 2009, Zhou and Zhu, 2011], we adapt an SVM dependency tree kernel model [Moschitti, 2006] for both tasks. The input to the SVM tree kernel model is a dependency path, created by the Stanford Parser [Marneffe and Manning, 2008]. While selecting relevant features from a parse tree for semantic tasks is difficult, SVM tree kernels avoid extracting explicit features from parse trees by calculating the inner product of the two trees. For the Binary task, our dependency path is the path between two entities. For the Triple task, the path is between entities and relation words, i.e., relation triples.

Tree kernels have been used in traditional RE and have helped achieve state of the art performance. But one challenge of using tree kernels on Open IE is that the lexicon of relations is much larger than those of traditional RE, making it difficult to include the lexical information as features. In this chapter I present our unlexicalized tree structure for Open IE, which is the first time an SVM tree kernel has been applied in Open IE. Experimental results on multiple datasets show our system outperforms state-of-the-art systems REVERB and OLLIE.

In addition to the supervised model, I also present an unsupervised model which relies on several heuristic rules. Results with this approach show that this simple unsupervised model provides a robust strong baseline for other

approaches[1].

3.1 Problem Definition and System Structure

The common definition of the Open IE task is a function from a sentence, s, to a set of triples, $\{< E1, R, E2 >\}$, where $E1$ and $E2$ are entities (noun phrases) and R is a textual fragment indicating a semantic relation between the two entities. Our "Triple task" is within this definition. However, it is often difficult to determine which textual fragments to extract. In addition, semantic relations can be implicit, e.g., the *located in* relation in the sentence fragment "Washington, US."

To illustrate how much information is lost when restricting the relation forms, we add another task, the "Binary task," determining if there is a relation between the two entities. It is a function from s, to a set of binary relations over entities, $\{< E1, E2 >\}$. This binary task is designed to overcome the disadvantage of current Open IE systems, which suffer because of restricting the relation form, e.g., to only verbs, or only nouns. These two tasks are independent of each other.

Figure 3.1 presents our Open IE system structure. Both tasks need preprocessing with the Stanford NLP tools (other equivalent tools such as Open NLP could be used). Entities and entity pairs within a certain token distance are extracted. Sentences are parsed with the Stanford Constituent parser [Klein and Manning, 2003] then transformed to dependency parse trees. We employ the typed collapsed dependency parse [Marneffe et al., 2006a], which is transferred from the constituent parsing and has proved to be useful for semantic tasks [MacCartney et al., 2006]. For the Binary task, an SVM model is employed to filter out the extracted entity pair candidates, and output pairs which have certain relations. For the Triple task, we identify relation word candidates of the pairs, based on POS regular expression patterns. Then another SVM model is employed to decide if the relation triples are correct or not.

[1]This work is published in NAACL 2013 [Xu et al., 2013].

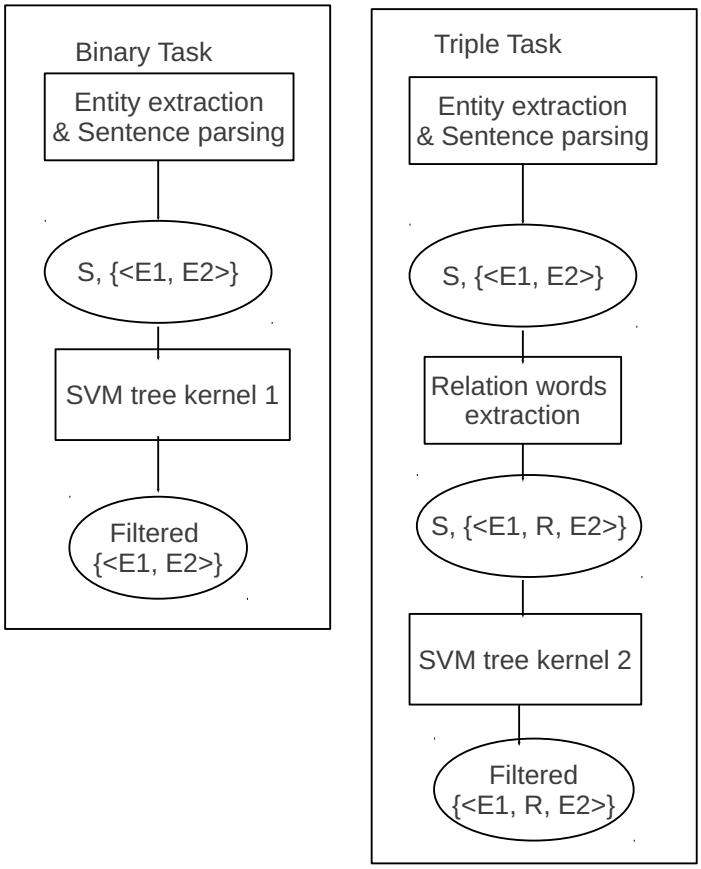


Figure 3.1: Our Open IE system structure.

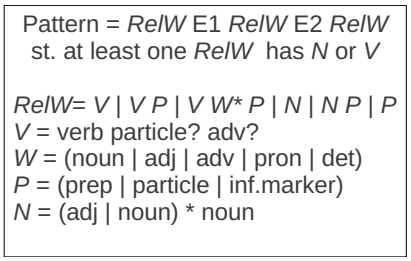


Figure 3.2: Relation Pattern Form (*RelW* represents *relation words*, E1 and E2 are two entities.)

3.2 Relation Candidate Extraction

For the Triple task, we extract textual fragments which match certain POS patterns in an entity pair's context as relation candidates for that pair. In our experiments, the fragments are n-grams with $n < 5$. Relation candidate words are between the entity pairs or in a window size of 10 before the first entity or after the second entity, which is experimentally a good choice to minimize noise while attaining maximum number of relations.

Our representation of POS regular expression pattern sets expands that of Fader et al. [2011]. The patterns are composed of verb and noun phrases (see Figure 3.2). A relation candidate can consist of words before, between, or after the pair, or the combination of two consecutive positions. Instead of extracting only verbal relations, e.g., *give birth to*, our patterns also extract relations specified through noun phrases. In the sentence "*Obama, the president of the United States, made a speech*," the relation *president* matches the relational form "RelW=N, N=noun." Our method can also extract relation words interspersed between the two entities: e.g., ORG *has* NUM *employees*, which matches the pattern "E1 RelW E2 RelW"; the first RelW matches V, with V=verb, and the second RelW matches N, with N=noun.

3.3 Tree Kernels

Many methods recognize the value of leveraging parsing information in support of semantic tasks. But selecting relevant features from a parse tree is a difficult task. SVM tree kernels avoid extracting explicit features from parse trees by

calculating the inner product of the two trees, so the tree kernel value depends on the common substructure of two trees. A tree kernel function over Tree T_1 and T_1 is:

$K(T_1, T_2) = \sum_{n1 \in N_{T_1}} \sum_{n2 \in N_{T_2}} \Delta(n_1, n_2)$,

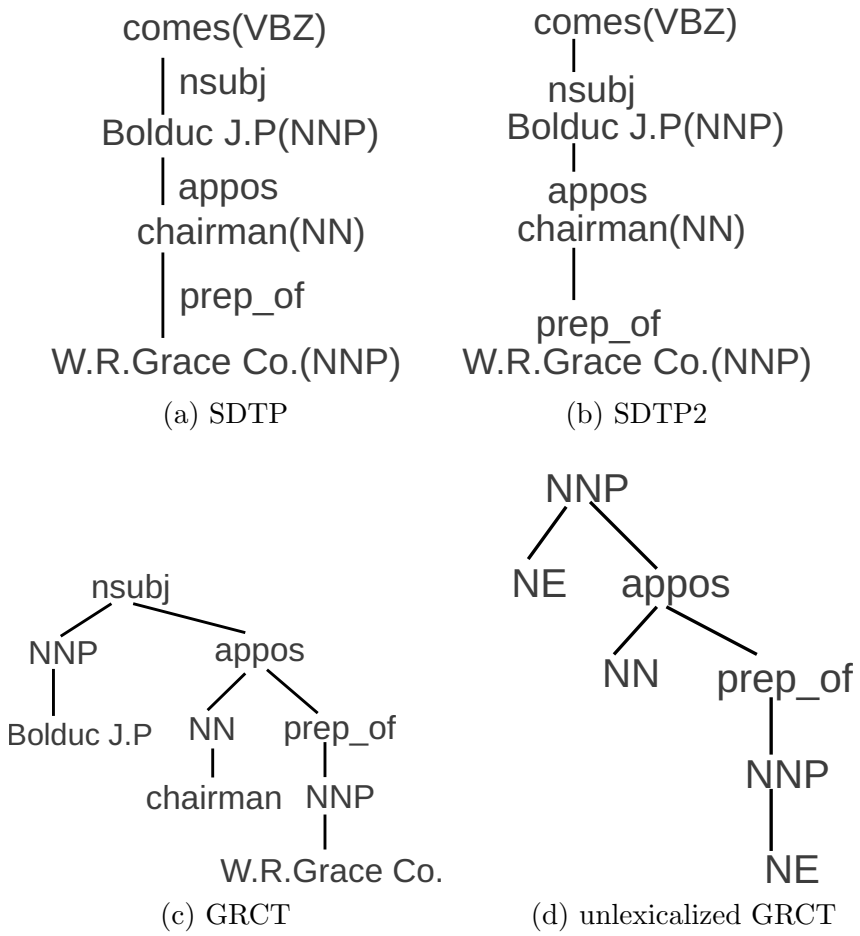
where N_{T_1} and N_{T_2} are the set of trees' nodes [Collins and Duffy, 2001]. The Δ function provides the basis for identifying subtrees of nodes, which is the essential distinction between different tree kernel functions. Here we adapt the partial tree kernel (PTK) proposed by Moschitti [2006][2], which can be used with both constituent and dependency parse trees. When the node labels of n_1 and n_2 are the same, the computation of Δ function of PTK is

$$\mu(\lambda^2 + \sum_{J_1, J_2, l(J_1)=l(J_2)} \lambda^{d(J_1)+d(J_2)} \prod_{i=1}^{l(J_1)} \Delta(c_{n_1}(J_{1i}), c_{n_2}(J_{2i}))); \quad (3.1)$$

$\Delta = 0$ when they are different. c_{n_1} and c_{n_2} are child sequences of nodes n_1 and n_2 respectively, $J_1 =< J_{11}, J_{12}, J_{13}... >$ and $J_2 =< J_{21}, J_{22}, J_{23}... >$ are index sequences of the two child sequences, J_{1i} and J_{2i} are the i-th children of the two sequences. $l()$ means the sequence length, $d(J_1) = J_{1K} - J_{11}$, where J_{1K} is the last index in the J_1 sequence. μ and λ are two decay factors for the height of the tree and the length of the child sequences respectively, which we choose the default setting in the experiments. For a more detailed description of PTK, please refer to Moschitti [2006].

Now we present our unlexicalized dependency tree structures for the tree kernel. One question arising in the conversion of dependency structures, such as the one in Figure 3.3a, for the tree kernel is how should we add POS tags and dependency link labels? The kernel cannot process labels on the edges; they must be associated with tree nodes. Our conversion is similar to the idea of a Grammatical Relation Centered Tree (GRCT) of Croce et al. [2011]. First we order the nodes of dependency trees so that the *dominant*, i.e., the parent of the dependency link is on the top, the *dependent*, i.e., the child is at the bottom. At this stage, the link label is with the corresponding *dependent* POS-tag and the word (Figure 3.3b). If a dominant has more than one child,

[2]Thanks to Prof. Moschitti for his PTK package.



(a) SDTP (b) SDTP2

(c) GRCT (d) unlexicalized GRCT

Figure 3.3: Example trees for shortest dependency path between *J.P. Bolduc* and *W.R.Grace Co.* in sentence "J.P. Bolduc, vice chairman of W.R.Grace Co., comes here." Figure (a) is the shortest dependency tree path (SDTP), (b) is the collapsed form, (c) is the GRCT, (d) is an unlexicalized GRCT with "NE".

the children will be ordered according to their position in the sentence, from left to right. Next, every node is expanded such that the *dependent* POS-tags are the children of the link labels and parent of their words. For example, in Figure 3.3c, *NN* is the child of *appos* and parent of *chairman*. It is on the left of *prep_of* because *chairman* is on the left of *W.R.Grace Co.* in the sentence. As customary in Open IE, we do not add content words, while function words are included to replace their POS tags. The unlexicalized GRCT is shown in Figure 3.3d. Note that for the root node, the link label is replaced by the POS-tag of the fist node in the path.

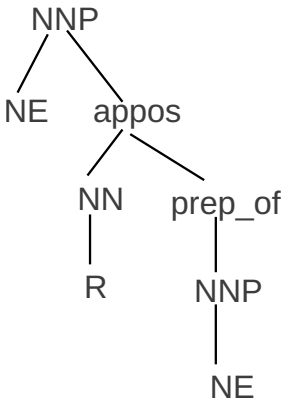
Recall that we have two tasks: detecting whether there is a relation between two entities (the Binary task), and whether the relation triple <*E1, relation, E2*> is correct (the Triplet task). We define two expanded versions of unlexicalized GRCT for these two tasks. The two versions contain different fragments of a dependency tree of a sentence.

For the Binary task, the shortest path between two entities' heads[3] is extracted and represented as a GRCT. The root node is the POS-tag of the first node in the path. "NE" is used to represent the position of two entities while relation words are not specified. Figure 3.3d shows the example final outcome of our tree structure. It is used to decide if there is a relation between the entities *Bolduc J.P.* and *W.R.Grace Co.*

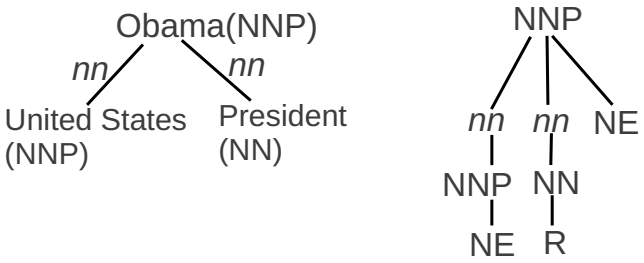
For the Triple task, we first extract relation words based on regular expression patterns as indicated in Section 3.2. If any relation word is between the shortest path of the two entities, the path is chosen as the input for SVM. Otherwise, two shortest paths between two entities and relation words will be extracted separately. The shortest path between one of the relation words and one of the entities will be attached to the path between two entities.

In our representation, relation words are tagged by having "R" as the child. Figure 3.4a shows the path form of the previous example. Figure 3.4b shows another example where "R" is not in the shortest path of the pair. The triple is <United States, president, Obama> for the sentence "*United States President Barack Obama says so.*" The figure on the left is the dependency path. The

[3]The head words of phrases are words which do not depend on any words in the phrases.



(a) Example 1.



(b) Example 2.

Figure 3.4: Tree structure with "R" added. Figure (a) is Example 1, which has R in the SDTP of the entity pair. Figure (b) is Example 2, with R not in the SDTP of the entity pair.

figure on the right is the final tree for the triple task. The root is the POS-tag for *Obama*.

For the Triple task we combine the tree kernel with a polynomial kernel [Moschitti, 2005] applied to a feature vector. A polynomial kernel is defined as:

$$K(x_1, x_2) = (x_1^T x_2 + c)^d, \tag{3.2}$$

where x_1 and x_2 are feature vectors of two instances, and d is the degree of the polynomial.

We designed a series of features to assist the tree kernel. The feature set is in Table 3.1. The third feature (F3) tries to preserve the semantic link between two discontinuous relation word segments. F6 constrains relation words to include only necessary prepositions. For verbal relations, if there is a preposition at the end of the relation word sequence, then there must be a preposition link

E feature	F1	the dependency link label between two entities, *null* if none.
R features	F2	whether relation is a noun phrase or a verb phrase
	F3	whether there is a link between the two segments (if there are two discontinuous segments)
between E and R	F4	whether there is a link between entities and the relation
	F5	the shortest dependency path distance between entities and the relation (1,2,3,4, or >4)
	F6	the preposition link and the last preposition word of relation (if there is such a link or word)
	F7	whether there is a conjunction link in the shortest path between entities and the relation
	F8	whether there is a apposition link in the shortest path between entities and the relation

Table 3.1: Noise filter feature vector.

between the relation and any of the two entities, and vice versa. For instance, in the sentence "*Bob teaches at the University,*" <Bob, *teach at*, University> is correct while <Bob, *teach*, University> is wrong. For nominal relations, inclusion of the head word is necessary. Prepositions can be ignored, but if they exist, they must match with the dependency link. We concentrate on verb prepositions because prepositions are more frequently attached to noun phrases than verb phrases. Verb relations have more preposition choices, and different choices have different semantic impact. For example, prepositions can indicate roles of entities, e.g., whether they are subjects or objects. But noun relations' prepositions are more fixed, such as *president of*. The last two features F7 and F8 are added according to the observation of experiment results in a development set: we notice that relation extraction errors arise frequently when there are apposition or conjunction structures between entities[4].

3.4 Unsupervised Method

We also propose the use of an unsupervised method based on heuristic rules to produce a relation candidate noise filter, i.e., selecting correct relations out of false relations, as an alternative to using SVM in the Triple task. The heuristic rules are also based on the Stanford collapsed dependency parsing. There are two parts in the noise filter: one is that the relation words should have necessary links with two entities and the other is that relation words should be consistent.

We first mention the heuristic rules for necessary dependency links. The intuition is from [Chan and Roth, 2011], they classified relations into 5 differ-

[4]However, according to the results on the test set, adding the two features does not solve the problem.

ent syntactic structures; premodifier, possessive, preposition, formulaic, and verbal. They proposed heuristic POS patterns covering the first four patterns with the exception of the verbal structure.

We present heuristic rules based on dependency paths instead of POS for the following structures, except the category formulaic.

- In a premodifier structure one entity and the relation are modifiers of the other entity, e.g., US. *President* Obama.
- In a possessive structure one entity is in a possessive case, e.g., Microsoft's *CEO* Steve Ballmer.
- In a preposition structure, relation words are related with one entity by a preposition, e.g., Steve Ballmer, *CEO* of Microsoft.
- In a verbal structure relations are verb phrases.

The heuristic rules are presented in Figure 3.5. The premodifier and possessive relation words are not along the dependency path between two entities[5]. When there is a direct dependency link between two entities that is labelled *nn* or *poss*, there should be an *nn* link between the second entity and the relation candidate (in Figure 3.5's top two rows). Otherwise, there should be links between the two entities and the relation, respectively (in Figure 3.5's last row). In this case, link types and directions are not constrained. For example, both *E1* ←(nsubj) *R* →(dobj) *E2* for the triple <Obama, visit, Canada> in "*Obama visited Canada.*" and *E1* →(appos) *R* →(prep_of) *E2* for the triple <Obama, president, United States> in "*Obama, the president of the United States, visited Canada.*" belong to that structure. To refine the verbal pattern, the link between the relation words and entities cannot be a conjunction.

Next, we need to check the consistency of relation words. Here consistency means whether the word sequence form a meaningful semantic unit. For example, *break the rule of* is consistent, while the candidate *contribution parents*

[5]At least for the Stanford dependency parser we used, it might be otherwise if another dependency parser is used.

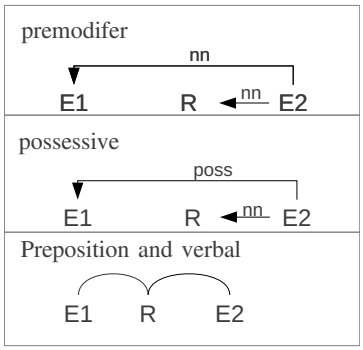


Figure 3.5: Dependent link heuristics for relation detection.

from *contribution from E1's parents* is not. Two separated sequences of relation words should have a dependency link between each other to confirm that they are semantically related. Relation sequences should include only necessary prepositions.

3.5 Experiments

We compared the unsupervised heuristic rule method and the supervised SVM method discussed above against REVERB [Fader et al., 2011] and OLLIE [Mausam et al., 2012], using three datasets. One dataset consists of sentences from the Penn Treebank, and the other two are the experiment datasets of REVERB and OLLIE respectively.

3.5.1 Treebank Set

Preparing Data

Within the research community, it is difficult to find Open IE test data which includes all kinds of relations. So we have created our own data from the Penn Treebank for evaluation[6]. The Penn Treebank contains sentences and their parse trees annotated by linguists. We assess the drop in Open IE performance introduced by using a syntactic parser compared to using "ideal" parse trees provided in the Penn Treebank. Named entities are tagged for every sentence using the Stanford NLP tool. Candidate NE pairs are extracted within a

[6]The data can be downloaded from `http://cs.ualberta.ca/~yx2/`.

certain distance[7]. We randomly selected 756 sentences from WSJ Sections 2-21 as our training set, 100 each from Section 22 and Section 23-24 as the develop and the test set, respectively. This is also the setting for most parsers.

We manually annotated whether there is a relation between two entities in a sentence (for evaluation of the Binary task). If there is a relation between two entities, the annotator needs to indicate which words are relation words (for evaluation of the Triple task). There is no restriction of relation forms for the annotator in this task.

After analyzing half of the relation instances in the training set, we notice that 28% of the relations are implicit relations, i.e., relations without words or with prepositions. Less than 1% are with adjectives, while 71% are noun or verb phrases. In those 71%, 60% are noun relations and 40% are verbal. The relation patterns from Section 3.2 can extract 80% of them. Our data contains more verbal relations than the ACE's RDC, but less than corpora used in other Open IE papers.

We evaluate each system by recall, precision, and F-score. The evaluation of the Binary task is based on entity pairs and is straightforward. The evaluation of the Triple task is based on relation triples. We need to manually compare the triples extracted by each system and the gold standard to avoid double-counting. For instance, if both *vice president* and *president* are extracted, it is counted as one[8]. Several entity pairs have multiple relations, such as "*A is CEO and founder of B.*" Any relation which can not be represented by a verb or noun is counted as one miss in the Triple task.

To compare with the REVERB system, NE pairs are labelled as two noun phrase chunks for the system input. It is difficult to compare with OLLIE, as the system is a black box with integrated entity extraction and parsing. We manually compared the pairs extracted by OLLIE and the tagged data. Only results of intersection entity pairs are considered. The threshold of OLLIE and REVERB confidence is set to achieve the best F-score in the development set.

[7]Here we set the distance as 20, determined by empirical evidence, a majority of the relations are within this distance.

[8]It is difficult to decide if *president* in this case is wrong. This is related to multi-word expression and will be future work.

	P	R	F-score
Treebank parsing + DP rules	0.833	0.549	0.662
Treebank parsing + SVM	0.896	0.767	0.826
Stanford parsing + DP rules	0.783	0.522	0.627
Stanford parsing + SVM	0.744	0.711	0.727
REVERB (no parsing)	0.333	0.1	0.153
OLLIE (MaltParser)	0.583	0.389	0.467

Table 3.2: Relation extraction results on Treebank set (Binary)

Results

The Binary task results on the test set are shown in Table 3.2. Each system decides whether there is a relation between two entities. The heuristic rule (DP rules) method, REVERB, and OLLIE each tag pairs containing a relation if any relation candidates are identified. As indicated, the SVM method performs the best, while the DP rules method ranked second. Note that OLLIE uses MaltParser, so it's better to compare with the coupling of SVM with Stanford Parser.

The Triple task results are shown in Table 3.3. We extracted 12084 relation candidates from our training sentences, i.e. our training size is 12084 for the SVM models. Each system extracts relation triples from sentences. The SVM features include both tree (Figure 3.4) and vector features (Table 3.1). *All relations* in the table include nominal, verbal, and implicit relations. To scrutinize the result, we also show the results on noun and verb relations separately. The SVM model achieves best performance, 33% improvement on nominal relation extractions over OLLIE. *Stanford parsing(also for pattern)+SVM* setting means that the relation candidate extraction is also based on the Stanford parsing result, while others' relation candidate extraction is based on the Treebank parsing. Comparing with *Stanford parsing + SVM*, we can see that the result decreases a little but is still better than OLLIE.

The loss of recall for systems (except SVM) in the Binary task can be explained by the fact that nearly 20% of relations are implicit.

In both the Binary and Triple tasks, one source of failure arose from conjunction and apposition structures. For example, in the sentence "*...industry*

All relations	P	R	F-score
Treebank parsing + DP rules	0.741	0.467	0.573
Treebank parsing + SVM	0.824	0.462	**0.592**
Stanford parsing + SVM	0.75	0.433	**0.549**
Stanford parsing(also for pattern)+SVM	0.7	0.43	**0.53**
OLLIE (MaltParser)	0.583	0.389	0.467
Noun relations	P	R	F-score
Treebank parsing + DP rules	0.75	0.735	0.742
Treebank parsing + SVM	0.829	0.708	**0.764**
Stanford parsing + SVM	0.756	0. 689	**0.721**
OLLIE (MaltParser)	0.8	0.408	0.54
Verb relations	P	R	F-score
Treebank parsing + DP rules	0.7	0.368	0.483
Treebank parsing + SVM	0.727	0.381	0.5
Stanford parsing + SVM	0.727	0.32	0.444
REVERB (no parsing)	0.286	0.381	0.327
OLLIE (MaltParser)	0.429	0.714	**0.536**

Table 3.3: Relation extraction results on Treebank set (Triple)

executives analyzed the appointment of the new chief executive, Robert Louis-Dreyfus, who joins Saatchi ..." the method can detect the relation <*chief executive*, *joins*, *Saatchi*>, but not <*Robert Louis-Dreyfus*, *joins*, *Saatchi*>. We attempted to address this problem by adding features into SVM linear kernel (Table 3.1), but the results in the test set show no success of the approach.

One cause of recall loss in the Triple task for REVERB and our two approaches is that verbal relation words can be non-consecutive. For instance, the preposition might be far away from the related verb in one sentence, in which case both our methods and REVERB can not confirm that extraction. OLLIE has better results on verb relations mainly because they use dependency link patterns to extract relation words, which alleviate the problem. On the other side, one drawback of OLLIE is that it failed to extract a few premodifer structure relations, e.g., "*U.S. President Obama.*" That may happen because they do not have an independent step for named entity extraction, which is crucial for that type of relation.

	P	R	F-score
Stanford parsing + DP rules	0.711	0.811	0.756
Stanford parsing + SVM	0.718	0.859	**0.781**
REVERB	0.577	0.95	0.716

Table 3.4: Relation extraction results on REVERB set (Triple).

3.5.2 REVERB Set

The authors of the REVERB method provided 1000 tagged training sentences and 500 test sentences. They also provide REVERB's extracted relation triples and the corresponding confidence for the 500 test sentences. The 500 test sentences are segmented into 5 folds for a significance t-test (Here we use the Student t-test.) At each iteration, the remaining 400 sentences are used as a development set to set the threshold of REVERB confidence.

To compare with REVERB, we use as input the sentences parsed by the Stanford parser and relation triples extracted by REVERB for both training and testing. The output of our system is true or false for every triple by using the tree kernel[9]. The SVM system is trained on the 1000 training sentences. The results are shown in Table 3.4. Only SVM is statistically significant better than REVERB (with $\alpha = 0.05$). Note that the results here seem better than the results shown on [Fader et al., 2011]. It is because our evaluation is based on the set REVERB extracted, while the results in [Fader et al., 2011] is based on the union relation set of several systems. We believe that this is sufficient to compare the three Open IE systems.

3.5.3 OLLIE set

The authors of the OLLIE system provide a test set which has 300 sentences and OLLIE extracted 900 triples. Our experiment setting on this dataset is similar to that of REVERB set. The SVM tree kernel model is trained on OLLIE's leave one out dataset. The results in Table 3.5 show our method achieves slightly better performance, although not statistically significant.

[9]The polynomial kernel is not used for REVERB and OLLIE data as the their relation word form is simpler than ours.

	P	R	F-score
Stanford parsing + SVM	0.685	0.941	**0.793**
OLLIE	0.667	0.961	0.787

Table 3.5: Relation extraction results on OLLIE set (Triple).

Besides errors caused by parsing, one main cause of loss of precision is that our system is unable to detect entities that are wrong as we only concern the head of the entity. For instance, "*Bogan 's Birmingham Busters , before moving to Los Angeles , California*" is one entity in one OLLIE relation, where only "*Bogan 's Birmingham Busters*" is the correct entity.

3.6 Summary

In this chapter I have described limitations of current Open IE systems, which concentrate on identifying explicit relations, i.e., relations which are mediated by open class words. This strategy ignores what we describe as implicit relations, e.g., *locate* relations in "*Washington, U.S.*"

We propose two subtasks for Open IE: first confirming whether there is a relation between two entities, and then whether a relation thus extracted is correct. The first task includes both implicit and explicit relations; the second task is common in the previous Open IE studies which deals with explicit relations. In our case we have developed an Open IE system which uses an SVM tree kernel applied to dependency parse trees for both tasks. Our system achieves superior results on several datasets. We also propose an unsupervised method which is based on heuristic rules from dependency parse links, and compared that with our SVM tree kernel methods. Our experiments show it is a strong baseline for Open IE.

The following chapters will describe systems to tackle problems we found in these experiments. We will improve our current model by including lexical information and extract implicit relations. For instance the relation *located* in "*Washington, U.S.*" will be extracted.

Chapter 4

A Lexicalized Tree Kernel for Open Information Extraction

Like most of the early Open IE systems, our tree kernel model from the previous chapter employs syntactic information such as parse trees and part of speech (POS) tags, but ignores lexical information. One major reason is that most Open IE training corpora contain only several thousands of sentences. If word features are included, there will be sparsity issue, i.e., supervised machine learning models can not estimate the weight (effect) of infrequent words correctly.

However, previous work suggests that Open IE would benefit from lexical information because the same syntactic structure may correspond to different relations. For instance, the relation triple <Annacone, coach of, Federer> is correct for the sentence "*Federer hired Annacone as a coach*", but not for the sentence "*Federer considered Annacone as a coach,*" even though they have the same dependency path structure [Mausam et al., 2012]. Lexical information is required to distinguish the two cases.

In this chapter I will present our second Open IE model, which incorporates a lexicalized tree kernel model that combines both syntactic and lexical information. In order to avoid lexical sparsity issues, we investigate two smoothing methods that use word vector representations: Brown clustering [Brown et al., 1992] and word embeddings created by a neural network model [Collobert and Weston, 2008]. To our knowledge, we are the first to apply word embeddings and to use lexicalized tree kernel models for Open IE.

Experiments on three datasets demonstrate that our Open IE system achieves absolute improvements in F-measure of up to 16% over the current state-of-the-art systems of Xu et al. [2013] and Mesquita et al. [2013]. In addition, we examine alternative approaches for including lexical information, and find that excluding named entities from the lexical information results in an improved F-score[1].

4.1 Related Work – Word Representation

Lexical features are one type of important feature in supervised models for natural language processing. They are usually represented as binary features: one binary feature represents one word and is set to 1 when the word exists, 0 if not. The number of these features is the size of vocabulary in the training set. However there is a sparsity problem with this feature type, as many words occur only once in the training set. Little can be learned with infrequent features and the supervised task. To alleviate the problem, researchers have proposed several approaches for representing words as word vectors, i.e., a fixed length of numerical vectors. The number of these features is determined by the selected length of the word vector.

Turian et al. [2010] categorized word representation techniques into 3 categories: distributional, clustering, and distributed representation. All these techniques follow one observation: words are identified by their context. The main difference lies in how they choose context and which technique to use to derive vectors.

Distributional word representation models learn word representation from a matrix: $W \times C$. In the matrix, each row represents a word and $|W|$ is the vocabulary size. Columns are context and $|C|$ is the context size. A model can reduce the word representation dimension with a certain function f that maps $W \times C$ to $W \times d$, where $|d| << |C|$. One representative model is Latent Semantic Analysis (LSA). In the model, a word's context is the documents which it occurs in. C's dimension is the size of the documents. In the matrix

[1]This research was previously published in [Xu et al., 2015].

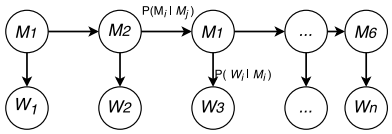


Figure 4.1: A bayesian network that defines the quality of clustering.

$W \times C$, if the word i is in the document j, then the matrix's (i, j) slot will be 1. The LSA model uses Singular Value Decomposition (SVD) for dimension reduction.

Clustering approaches take the matrix $W \times C$ and cluster based on every row. One representative is the Brown Clustering model [Brown et al., 1992], which is a bottom-up hierarchical clustering model for words. The input is a sequence of words, while the output is a binary tree with words as leaves. Liang [2005] proposed an algorithm that reduce the time cost from $O(|V|^3)$ to $O(|V|k^2 + n)$, where $|V|$ is the dictionary size, k is the clustering size, and n is the corpus length. Following is a brief description of the algorithm:

1. The algorithm starts with k most frequent words, each in their own cluster.

2. Loop: the next most frequent words, $k + 1$, $k + 2$,.... is added.

3. Now we choose two clusters i and j, and merge them. We choose the merge that gives a maximum value of the current clustering: $Quality(Clustering)$.

4. Do another (k-1) merges, to create one final cluster.

The quality of the merge is defined as the sum of mutual information of two adjacent clusters. Figure 4.1 gives the idea of adjacency of clusters. The probability of a cluster depends on its previous cluster, and the probability of a word depends on its cluster.

Distributed models are neural network models. They have been revived in the last several years and achieve state-of-the-art results in many NLP tasks. The induced representations are called *word embeddings*.

One representative distributed model is the word2vec model [Mikolov et al.,

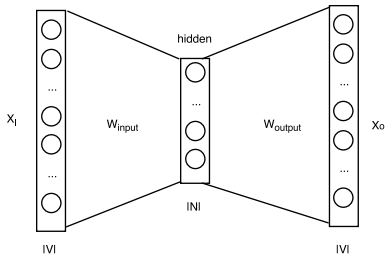


Figure 4.2: The skip-gram word2vec model.

2013]. Its skip-gram model includes one hidden layer and a softmax function to represent the probability $p(c_i|w_j)$, where c_i is a context word of w_j.

Figure 4.2 shows the simplified skip-gram model structure which uses only one context word (You can choose $2n$ context words, where n on the left side of the word in a sentence and n on the right side.) The input vector indicates which word is the input. If the input is w_j then only j slot is 1, others are 0. The first weight matrix W_{input} is the input word embedding, where each row represents each word. Thus the hidden layer h becomes the input word vector. The second weight matrix is the output word embedding. The output is the probability of every word as the context of the input word. Note that in this model, each word has two word representations, one is the input word embedding, the other is the output word embedding. In reality researchers use the input word embeddings as features for other NLP tasks.

Mikolov et al. [2013] proposed to use the objective function in Equation 4.1 to train the model.

$$E = -log\sigma((v'_{c_i})^T h) - \sum_{c_k \in W_{neg}} log\sigma(-(v'_{c_k})^T h) \tag{4.1}$$

where v' represents a vector in the output weight column, W_{neg} are sampled negative examples of words that are not in the context of input words, and σ is a sigmoid function:

$$y = \frac{1}{1 + e^{-x}} \tag{4.2}$$

According to the function 4.1, each backpropagation will affect the vectors of the input word, the context word, and the sampled negative words, which

saves a significant amount of time in training (For more explanations, see [Rong, 2014].)

Word embedding vectors created by neural network models have been used not only on deep learning models [Socher et al., 2012, 2013] but also on other models. For example, Turian et al. [2010] proposed a conditional random field (CRF) model with word embeddings as features for named entity extractions. Kuksa et al. [2010] created a string kernel model with word imbeddings for Bio-relation extraction. Motivated by these studies, our SVM tree kernel incorporates Gaussian similarity on word embedding vectors, which are created by a neural network model.

4.2 System Architecture

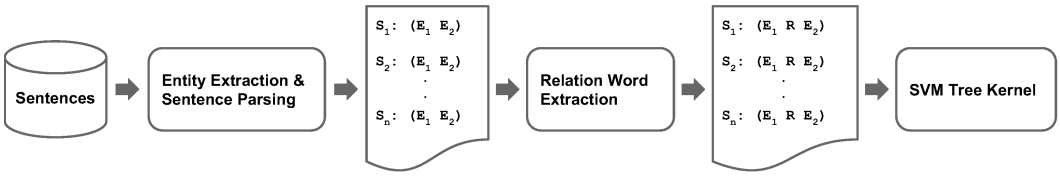


Figure 4.3: Our Open IE system structure.

Our system structure is similar to our first tree kernel-based model. It consists of three modules: entity extraction, relation candidate extraction, and tree kernel filtering. The system structure is outlined in Figure 4.3. We identify named entities, parse sentences, and convert constituency trees into dependency structures using the Stanford tools [Manning et al., 2014]. Entities within a fixed token distance (set to 20 according to the observation on the development set) are extracted as pairs $\{< E_1, E_2 >\}$. We then identify relation candidates R for each entity pair in a sentence, using dependency paths. Finally, the candidate triples $\{< E_1, R, E_2 >\}$ are paired with their corresponding tree structures, and provided as input to the SVM tree kernel. Our Open IE system outputs the triples that are classified as positive. In the following sections, we describe the components of the system in more detail.

4.3 Relation Candidates

Relation candidates are words that may represent a relation between two entities. We consider only lemmatized nouns, verbs and adjectives that are within two dependency links from either of the entities. Following Wu and Weld [2010] and Mausam et al. [2012], we use dependency patterns rather than POS patterns, which allows us to identify relation candidates which are farther away from entities in terms of token distance. This is different from our last work in [Xu et al., 2013], which uses part of speech patterns. From that work we notice that using POS patterns creates too much noise, i.e., creating too many poor relation candidates.

To extract relation candidates we first extract the head words for the two entities. We then extract the first two content words along the dependency path between the head words of E_1 and E_2. In the following example (Figure 4.4), the path is $E_1 \leftarrow$ encounter $\rightarrow$ build $\rightarrow E_2$, and the two relation word candidates between "Mr. Wathen" and "Plant Security Service" are *encounter* and *build*, of which the latter is the correct one.

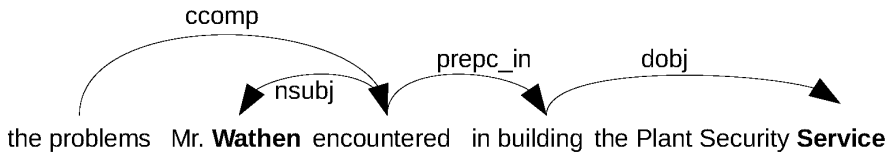


Figure 4.4: One relation candidate example where the relation word is on the dependency path between the two entities.

If there are no content words on the dependency path between the two entities, we instead consider words that are directly linked to either of them. In the following example (Figure 4.5), the only relation candidate is the word *battle*, which is directly linked to "Edelman."

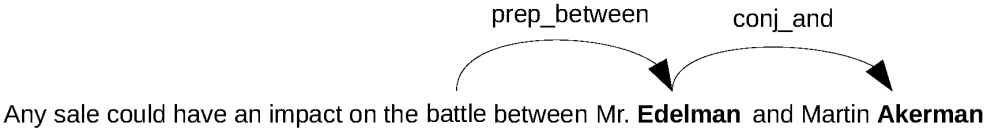
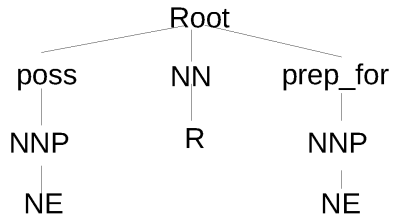


Figure 4.5: One relation candidate example where the relation word is not on the dependency path between the two entities.

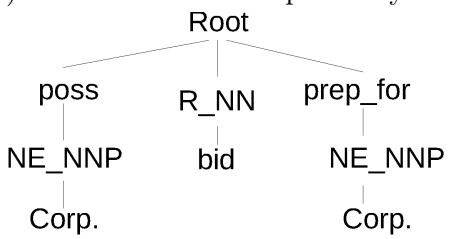
The relation candidates are manually annotated as correct or incorrect in the training data for the tree kernel models described in the following section.

4.4 Lexicalized Tree Kernel

We use a supervised lexicalized tree kernel to filter negative relation candidates from the results of the candidate extraction module. For semantic tasks, the design of input structures to tree kernels is as important as the design of the tree kernels themselves. In this section, we introduce our tree structure, describe the prior basic tree kernel, and finally present our lexicalized tree kernel function.



(a) An un-lexicalized dependency tree.



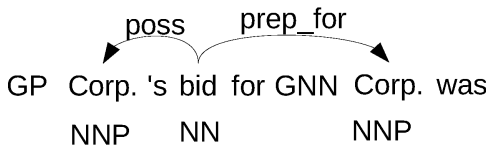
(b) A lexicalized dependency tree.

Figure 4.6: An unlexicalized tree and the corresponding lexicalized tree.

4.4.1 Tree Structure

In order to formulate the input for tree kernel models, we need to convert the dependency path to a tree-like structure with unlabelled edges. The target dependency path is the shortest path that includes the triple and other content words along the path. Consider the following example, which is a simplified representation of the sentence *"Georgia-Pacific Corp.'s unsolicited $3.9 billion bid for Great Northern Nekoosa Corp. was hailed by Wall Street."*

The candidate triple identified by the relation candidate extraction module is *<Georgia-Pacific Corp., bid, Great Northern Nekoosa Corp.>*.



Our unlexicalized tree representation model is similar to the unlexicalized representations in Chapter 3, except that instead of using the POS tag of the path's head word as the root, we create an abstract *Root* node. We preserve the dependency labels, POS tags, and entity information as tree nodes: (a) the top dependency labels are included as children of the abstract *Root* node, other labels are attached to the corresponding parent labels; (b) the POS tag of the head word of the dependence path is a child of the *Root*; (c) other POS tags are attached as children of the dependency labels; and (d) the relation tag 'R' and the entity tags 'NE' are the terminal nodes attached to their respective POS tags. Figure 4.6(a) shows the unlexicalized dependency tree for our example sentence.

Our lexicalized tree representation is derived from the unlexicalized representation by attaching words as terminal nodes. In order to reduce the number of nodes, we collapse the relation and entity tags with their corresponding POS tags[2]. Figure 4.6(b) shows the resulting tree for the example sentence.

4.4.2 Tree Kernels

Tree kernel models extract features from parse trees by comparing pairs of tree structures. The essential distinction between different tree kernel functions is the Δ function that calculates similarity of subtrees. Our modified kernel is based on the SubSet Tree (SST) Kernel proposed by Collins and Duffy [2002]. What follows is a simplified description of the kernel; a more detailed description can be found in the original paper.

[2]We haven't investigated into the effect of collapse yet, which will be our future work.

The general function for a tree kernel model over trees T_1 and T_2 is:

$$K(T_1, T_2) = \sum_{n_1 \in T_1} \sum_{n_2 \in T_2} \Delta(n_1, n_2), \tag{4.3}$$

where n_1 and n_2 are tree nodes. The Δ function of SST kernel is defined recursively:

1. $\Delta(n_1, n_2) = 0$ if the productions (context-free rules) of n_1 and n_2 are different.

2. Otherwise, $\Delta(n_1, n_2) = 1$ if n_1 and n_2 are matching pre-terminals (POS tags).

3. Otherwise,
   $\Delta(n_1, n_2) = \prod_j [1 + \Delta(c(n_1, j), c(n_2, j))]$,

where $c(n, j)$ is the jth child of n.

4.4.3 Lexicalized Tree Kernel

Since simply adding words to lexicalize a tree kernel leads to sparsity problems, a type of smoothing must be applied. Croce et al. [2011] employed word vectors created by Singular Value Decomposition [Golub and Kahan., 1965] from a word co-occurrence matrix. Plank and Moschitti [2013] used word vectors created by the Brown clustering algorithm [Brown et al., 1992]. Srivastava et al. [2013] also used word embeddings created by [Collobert and Weston, 2008] but their tree kernel did not incorporate POS tags or dependency labels.

We propose using word embeddings created by a neural network model [Collobert and Weston, 2008], in which words are represented by n-dimensional real valued vectors. Each dimension represents a latent feature of the word that reflects its semantic and syntactic properties. Next, we describe how we embed these vectors into tree kernels.

Our lexicalized tree kernel model is the same as SST, except in the following case: if n_1 and n_2 are matching pre-terminals (POS tags), then

$$\Delta(n_1, n_2) = 1 + G(c(n_1), c(n_2)), \tag{4.4}$$

where $c(n)$ denotes the word w that is the unique child of n, and $G(w_1, w_2) = exp(-\gamma\|w_1 - w_2\|^2)$ is a Gaussian function for two word vectors, which is a valid kernel. A function is a valid kernel function if it satisfies two properties: symmetry and positive semi-definite.

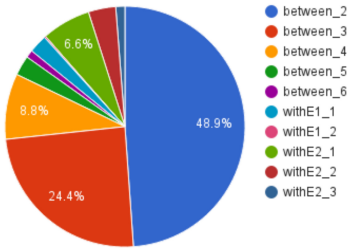
We examine the contribution of different types of words by comparing three methods of including lexical information: (1) relation words only; (2) all words (relation words, named entities, and other words along the dependency path fragment); and (3) all words, except named entities. The words that are excluded are assumed to be different; for example, in the third method, $G(E_1, E_2)$ is always zero, even if $E_1 = E_2$.

4.5 Experiments

In this section, we evaluate alternative tree kernel configurations, and compare our Open IE system to previous work. Our metrics are precision, recall, and F_1 score.

We perform experiments on three datasets (Table 4.1): the Penn Treebank set [Xu et al., 2013], the New York Times set [Mesquita et al., 2013], and the ClueWeb set which we created for this project from a large collection of web pages. Note that we revised annotation on several sentences on the Treebank dataset since 2013, which is the main reason that the result of [Xu et al., 2013] in Table 4.3 is different from the one in the last chapter (The result in Table 4.3 is from an unlexicalized model that we retrained the based on the new annotation.)

One advantage of using dependency path patterns for relation candidate extraction is that the candidate size is decreased dramatically compared with using POS patterns. Note that the relation candidate size in the training data is the training size of the SVM models. In the data of 2013, we have 12,084 relation triple candidates for 759 sentences and 2086 candidate entity pairs; while in the current Treebank data, we have only 3,235 relation triple candidates (We set the threshold of dependency path distance between two entities as 6). However, the performance is increased dramatically.



type	frequency
between_2	238
between_3	119
between_4	43
between_5	13
between_6	5
withE1_1	12
withE1_2	1
withE2_1	32
withE2_2	18
withE2_3	6

Figure 4.7: The frequency distribution of dependency path distance between relations and entities.

Figure 4.7 shows the frequency distribution of dependency path distance between gold standard relation head words and entities on the treebank training data when using the gold standard parse trees. *between_2* means the relation head word is on the dependency path of the two entities where the path has two links, *withE1_1* means that the relation word is not on the path and is connected with entity 1 with distance 1, and *withE2_1* means that the relation word is connected with entity 2 with distance 1. We can see that 82.1% relations are on the dependency path between two entities that is no longer than four. Most of the relations on the side are connected directly with entity 2. In our experiments, we set the threshold of *between* distance as 6 and *side* distance as 1 on the treebank set, while we set the threshold of *between* distance as 4 in ClueWeb and NewYork datasets because of the noise in parsing web page text.

After relation candidate extraction, we train SVM models on the Penn Treebank training set and tested on the three test sets, which makes the Penn Treebank set as in-domain, while the other two sets out-of-domain. For word embedding and Brown clustering representations, we use the data provided by Turian et al. [2010]. The SVM parameters, as well as the Brown cluster size and code length, are tuned on the development set.

Table 4.2 shows the effect of different smoothing and lexicalization techniques on the tree kernels. In order to focus on tree kernel functions, we use

Set	train	dev	test
Penn Treebank	759	100	100
New York Times	—	300	500
ClueWeb	—	450	250

Table 4.1: Data sets and their size (number of sentences).

Smoothing	Lexical info	P	R	F_1
none (Xu13)	none	85.7	72.7	78.7
none	all words	89.8	66.7	76.5
Brown (PM13)	relation only	88.7	71.2	79.0
Brown (PM13)	all words	84.5	74.2	79.0
Brown (PM13)	excl. entities	86.2	75.8	80.7
embedding	relation only	93.9	69.7	80.0
embedding	all words	93.8	68.2	79.0
embedding	excl. entities	95.9	71.2	**81.7**

Table 4.2: The results of relation extraction with alternative smoothing and lexicalization techniques on the Penn Treebank set (with our relation candidate extraction and tree structure).

the same relation candidate extraction (Section 4.3) and tree structure (Section 4.4.1) for different tree kernel models. The results in the first two rows indicate that adding unsmoothed lexical information to the tree structure in Chapter 3 is not helpful, which we attribute to data sparsity. On the other hand, smoothed word representations do improve F-measure. Surprisingly, a neural network approach of creating word embeddings actually achieves a lower recall than the method of [Plank and Moschitti, 2013] that uses Brown clustering; the difference in F-measure is not statistically significant according to compute-intensive randomization test [Padó, 2006].

With regards to lexicalization, the inclusion of relation words is important. However, unlike [Plank and Moschitti, 2013], we found that it is better to exclude the lexical information of entities themselves, which confirms the findings of [Riedel et al., 2013]. We hypothesize that the correctness of a relation triple in Open IE is not closely related to entities. Consider the example mentioned in [Riedel et al., 2013]: for relations like "*X visits Y*", X could be a person or organization, and Y could be a location, organization, or person.

Our final set of experiments evaluates the best-performing version of our

system (the last row in Table 4.2) against two state-of-the-art Open IE systems: Mesquita et al. [2013], which is based on several hand-crafted dependency patterns; and the one in Chapter 3, which uses POS-based relation candidate extraction and an unlexicalized tree kernel. Tree kernel systems are all trained on the Penn Treebank training set, and are tuned based on the corresponding development sets.

The results in Table 4.3 show that our system consistently outperforms the other two systems, with absolute gains in F-score between 4 and 16%. We include the reported results of [Mesquita et al., 2013] on the New York Times set. The ClueWeb results were obtained by running the respective systems on the test set, except that we used our relation candidate extraction method for the unlexicalized tree kernel of [Xu et al., 2013].

We conclude that the substantial improvement on the Penn Treebank set can be partly attributed to a superior tree kernel, and not only to a better relation candidate extraction method. Another observation we can make is that the Brown clustering and word embedding are also comparable in terms of the final relation extraction results.

The ClueWeb set is quite challenging because it contains web pages which can be quite noisy. As a result we've found that a number of Open IE errors are caused by parsing. Conjunction structures are especially difficult for both parsing and relation extraction. For example, our system extracts the relation triple <*Scotland, base, Scott*> from the sentence "*Set in 17th century Scotland and based on a novel by Sir Walter Scott, its high drama...*" with the wrong dependency path *Scotland* $\overset{conj\_and}{\rightarrow}$ *based* $\overset{prep\_by}{\rightarrow}$ *Scott*.

Adding additional information from context words that are not on the dependency path between two entities may alleviate this problem, which we intend to address in the future.

4.6 Summary

This chapter presented our lexicalized tree kernel model for Open IE, which incorporates word embeddings learned from a neural network model. Our

	P	R	F_1
	Penn Treebank set		
Xu et al. [2013]*	66.1	50.7	57.4
Brown (PM13)	82.8	65.8	73.3
Ours (embedding)	91.8	61.6	**73.8**
	New York Times set		
Mesquita et al. [2013]*	72.8	39.3	51.1
Brown (PM13)	83.5	44.0	**57.6**
Ours (embedding)	85.9	40.7	55.2
	ClueWeb set		
Xu et al. [2013]	54.3	35.8	43.2
Mesquita et al. [2013]	63.3	29.2	40.0
Brown (PM13)	54.1	31.1	39.5
Ours (embedding)	45.8	51.9	**48.7**

Table 4.3: Comparison of complete Open IE systems. The asterisks denote results reported in previous work.

system combines a dependency-based relation candidate extraction method with a lexicalized tree kernel, and achieves state-of-the-art results on three datasets. Our experiments on different configurations of the smoothing and lexicalization techniques show that excluding named entity information is a better strategy for Open IE.

Chapter 5

Implicit Relations in Nested Named Entity

Relation extraction, whether traditional relation extraction, *e.g.*, [Chan and Roth, 2011, Plank and Moschitti, 2013], or open information extraction, *e.g.*, [Banko et al., 2007, Xu et al., 2013], focuses on extracting relations between two named entities. Most of these studies concentrate on two entities that are neither consecutive nor overlapping and the relations are explicitly mentioned by words or phrases. For example in the sentence "*Bob teaches at the University.*" the relation between *Bob* and *the University* is explicitly represented by **teach**.

Frequently, entities are nested within each other in text. These entities are called nested named entities. Examples are *Germany's people*, *Google Images*, and *Shakespeare's book*. There are relations implied in these structures. For example, *Germany's people* **live** in *Germany*, *Google Images* is **owned** by *Google*, *Shakespeare's book* is **written** by *Shakespeare*. Most current relation extraction systems will ignore these implicit relations. Extracting the implicit relations in nested named entities is not only an interesting linguistic task but also useful for NLP applications such as question answering and textual entailment.

Unfortunately, nested named entities have not been annotated in the widely used evaluation corpora for named entity recognition (NER) such as CoNLL, MUC-6, and MUC-7 NER, let alone implicit relations between them. There are few NER models that can handle nested named entities. As noted by

Finkel and Manning [2009] this has largely been for practical, not ideological, reasons. Luckily there is a surging interest in this topic, for example, in the entity linking field researchers have been debating about how to annotate nested named entities, and models have been proposed on how to extract them [Lu and Roth, 2015].

In this chapter, we concentrate on extracting implicit relations between nested named entities. We begin our study by first creating a dataset of nested named entities and their corresponding relations. To avoid manual work, which is time consuming, we extract nested named entities from Wikipedia definition sentences such as *Google Images is a search service owned by Google*, where named entities such as *Google* are already tagged in the web page. We create two sets of relations as target relation sets. One is a natural language relation (NL-rel) set, which is also from the definition sentences, e.g., **own** in the previous example. The other is a Freebase relation (FB-rel) set, which leverages the links between Wikipedia and Freebase, a large community-curated database of well-known people, places, and things (entities).

Using our datasets, we construct a feature-based model for automatic implicit relation extraction. To the best of our knowledge, we are the first to study this problem systematically.

5.1 Related Work

There are several fields that are closely related to implicit relation extraction for nested named entities.

One is the semantic interpretation of noun compounds, which has been a long-standing area of interest in NLP research. It can either be defined as a semantic classification problem or a paraphrase problem. For the former, researchers propose a set of relations, and the implicit relations of noun-noun compounds are identified by supervised machine learning models [Girju et al., 2005, Rosario and Hearst, 2001]. For the latter, the task is to come up with verbs for the relations. For example, *honey bee* has *produce* relation and *apple pie* has *made of* relation. Our work belongs to the first category. However,

we concentrate on noun phrases that contain named entities and, instead of manual labelling, we generate training and testing data automatically from Wikipedia. As a result, our dataset has ten thousands of instances instead of hundreds.

There are scattered studies on implicit relations in nested named entities. A similar phenomenon is semantic relations expressed by prepositions. For example, *a dessert to die for* implies a *purpose* relation between *dessert* and *die*; and *her death from pneumonia* expresses a *cause* relation. Hovy et al. [2010] and Srikumar and Roth [2013] proposed an inventory of relations and built classification models to predict the relations in the preposition structure. However, they did not concentrate on nested named entities. The arguments of prepositions, i.e., the governor and the object, are not limited to nouns.

Peñas and Ovchinnikova [2012] are among the first to attempt to study the relations implied in nested named entities. Their model depends on a relation triple database and the existence of evidence in the corpus to identify the types of entities. For example, to extract the relation in *GTech Images* there should be sentences such as "*GTech Images is a product...*" However this evidence may not exist. We agree with them in that the implicit relations are dependent upon the entity types. However, we wish to take a further step, using features such as the head of entity mentions, entity shapes, and entity embeddings to represent the latent type instead of explicitly identifying the type. Besides their study is based on 77 instances, while ours is based on thousands.

5.2 Data annotation and analysis

5.2.1 Data annotation

Few information extraction corpora have taken nested named entities into consideration. In order to study the phenomenon systematically we create an automatically annotated corpus from Wikipedia definition sentences. The corpus contains pairs of entities, with one entity (sub-NE) as the substring of the other (super-NE), and their relation(s)[1]. In the following, we will describe

[1]We intend to release this data online.

our automatic annotation process.

One Wikipedia page is usually about one entity, with definition sentences of the entity at the top. The advantage of Wikipedia definition sentences is that no named entity recognition is needed. We assume that the first sentence of every page is a definition sentence and its subject is the title entity, even if the word sequences do not match. Other named entities are the ones with links to other Wikipedia pages. Among the definition sentences, we choose the ones which contain non-subject entities that are substrings of the titles. Following is one example, a definition sentence from the page with the title *Isle of Wight Festival*.

(1) The Isle of Wight Festival is a music festival which takes place every year on the [[Isle of Wight]] in [[England]].

Here the double square brackets represent linked word sequences. *Isle of Wight* is linked to another page about the *Isle of Wight County*. It is chosen as the sub-NE because it is contained in the title. The substring entity and the title are our target nested named entity pair.

There are cases when the title contains brackets, such as *Invisible (Jaded Era song)* and *Gary Locke (Scottish footballer)*. We consider them as two entities and choose the one that contains the sub-NE as the super-NE. For the two example titles, the super-NEs will be *Jaded Era song* and *Scottish footballer*.

After the entity pair identification, we extract the relations between the entities with two approaches. The first is extracting a natural language relation (NL-rel) from the definition sentence. We use the Stanford typed dependency parser [Marneffe et al., 2006b] to parse the sentence. After identifying the head word of each entity, we extract the verb on the shortest dependency path between the head words of the subject and the sub-NE as the relation. In Example (1), the head of the subject is *Festival* according to the dependency path. The head for the second entity (sub-NE) is *Isle*. The shortest dependency path between the two words is: *Festival* $\overset{nsubj}{\leftarrow}$ *festival* $\overset{rcmod}{\rightarrow}$ *take* $\overset{prep\_on}{\rightarrow}$

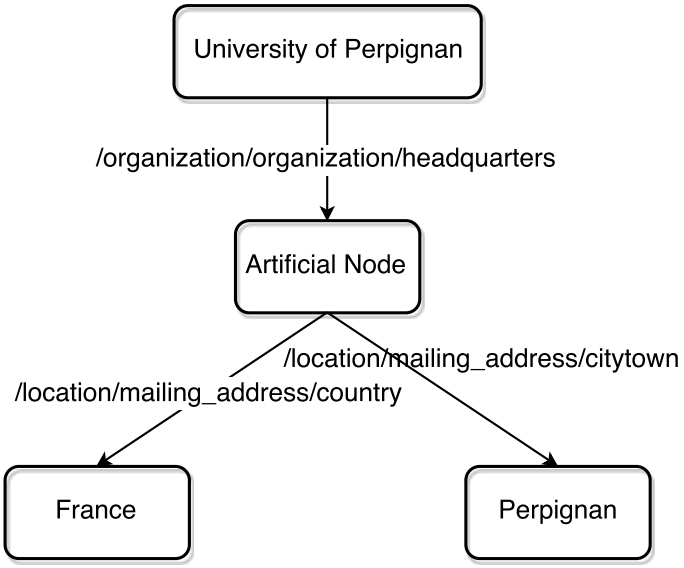


Figure 5.1: One indirect relation example in Freebase. The relation between *University of Perpignan* and *Perpignan* has two links.

Isle. We extract *take place* as the NL-rel for the nested named entity pair[2].

The second approach is to extract a Freebase relation (FB-rel). First we extract the Freebase ID for every Wikipedia entity. Here we use a Freebase dump file, which represents Freebase as a set of triples of the form <left argument, relation, right argument>. We extract any left argument of relation */type/object/key* when the right argument starts with */wikipedia/en_id/*. This left argument will be the Freebase ID and the right argument's suffix after */wikipedia/en_id/* will be the Wikipedia page title. For example, if the right argument is */wikipedia/en_id/Langenstein_Castle*, the Wikipedia page title is likely to be *Langenstein Castle*.

After the Freebase ID extraction, we extract Freebase relations between the nested named entities. We use both the Freebase dump file and the graph representation file of Freebase from [Yao and Van Durme, 2014]. The Freebase dump file has direct link relations between two entity IDs, while the graph representation file has relations with two hops, such as the relation */organization/organization/headquarters␣ /location/mailing_address/citytown* between *University of Perpignan* and *Perpignan* as in Figure 5.1.

[2]We extract objects of the light verb *take* to form a relation phrase.

	train	dev	test
NL-rel set	17659	5786	4635
NL-rel set with Freebase IDs	13344	4040	3329
FB-rel set	4070	1169	413

Table 5.1: Size of the dataset. Row 1 shows the size of entity pairs with NL-rels. Row 2 represents the size of a subset of Row 1, where entities have Freebase IDs. Row 3 shows the size of a subset of Row 2, where entities have Freebase relation(s).

We take relations' direction into account. For example there is a relation */location/location/contains* from *Vernon* to *Vernon College*, so we add a hyphen to the relation and use *-/location/location/contains* to represent the relation from *Vernon College* to *Vernon*. Note that one nested named entity pair might have several Freebase relations.

5.2.2 Annotation analysis

DataSet Size

Table 5.1 shows the nested named entity pair set size. The first row shows the size of entity pairs with NL-rels. The second row shows the number of pairs in the NL-rel set that have Freebase IDs for both super-NEs and sub-NEs. The third row shows the number of pairs in the NL-rel set that have Freebase relation(s).

Note that the size of the NL-rel set is larger than the FB-rel set. There is a decrease from Row 1 to Row 2 as we did not find Freebase IDs for a few Wikipedia entities because we can not find sub-NEs' Wikipedia pages. There is a larger decrease from Row 2 to Row 3 because of the well known Freebase incomplete problem. It is possible that relations are not tagged in Freebase even though they exist. For example, there should be a */sports/sports_team/location* relation between *South Fremantle Football Club* and *Fremantle*, which is missing from the version of Freebase dump we downloaded (It is the version of Sep. 28, 2014.)

Relation Distribution

Here we consider the implicit relation extraction as a multi-class classification task with frequent relations as classes.

For the NL-rel set, we extract the relations that occur with more than 40 nested entity pairs, and categorize others into one class *Other*. In total, there are 48 relation verbs: {*base, build, comprise, create, design, develop, establish, headquarter, hold, host, locate, make, manage, manufacture, produce, ...* }. Originally these 48 relations cover around 67% of the nested named entity pairs in the training set. To improve the classification models' recall, we sample only 10% of *Other* relations in the training set while keeping the development and test set untouched.

For the FB-rel set, we also extract frequent relations, *i.e.*, relations that occur more than thirty times. There are 38 such relations. Then we notice that a lot of relations are symmetric relations, such as *-/music/artist/album* and */music/album/artist*. Certain relations are highly correlated, such as */military/military_conflict/locations* with *-/location/location/events*. We calculate the pointwise mutual information (PMI) between relations R_1 and R_2:

$$PMI(R_1, R_2) = \log \frac{N * NEPair_1 \cap NEPair_2}{NEPair_1 \cup NEPair_2} \tag{5.1}$$

where N is the number of NE pairs in the training set and $NEPair_i$ is the number of pairs with R_i. We then merge the relations manually, guided by the PMI value with the more frequent relation as the final relation for a class. Finally there remains 18 relations and one *Other* class.

Table 5.2 shows the Freebase relation set and their frequency in the training set. Note that there are two compound relations here, e.g., */organization/organization/headquarters␣ /location/mailing_address/citytown*. Compound relations mean that the entities are not directly linked but have an artificial node between the two entities (Figure 5.1). From the table we can see that these frequent relations cover around 90% of the nested named entities in the FB-rel set.

One question is whether the high coverage still holds in the set of nested named entity pairs whose relations are missing in Freebase? To show that

Freebase relations	frequency	Precision	Recall	F-score
-/music/artist/album	939	95.5	93.7	94.6
-/location/location/contains	1146	85.8	73.0	78.9
-/location/location/nearby_airports	556	99.1	98.2	98.7
/sports/sports_team/location	260	95.8	85.2	90.2
-/location/location/events	202	92.9	89.7	91.2
-/symbols/name_source/namesakes	138	83.8	50.8	63.3
-/music/composer/compositions	129	93.5	100	96.6
/aviation/aircraft_model/manufacturer	127	100	95.8	97.9
/organization/organization/headquarters /location/mailing_address/citytown	97	25.0	5.0	8.3
-/travel/travel_destination/tourist_attractions	58	71.4	35.7	47.6
-/location/location/people_born_here	48	100	15.8	27.3
-/tv/tv_actor/starring_roles /tv/regular_tv_appearance/series	82	92.3	85.7	88.9
-/organization/organization_founder/organizations_founded	43	77.8	41.2	53.8
-/newspaper_circulation_area/newspapers	35	71.4	100	83.3
-/biology/organism_classification/lower_classifications	40	100.0	50.0	66.7
-/base/cars_refactor/make/model_s	30	100.0	25.0	40.0
/user/robert/mobile_phones/mobile_phone/brand	33	100.0	75.0	85.7
-/computer/software_developer/software	30	80.0	57.1	66.7
Other	418			
weighted average F-score		85.1		

Table 5.2: Freebase Relation Set. Column 2 shows the FB-rels frequency distribution in the training set. Column 3, 4, and 5 show the performance of the neural network model (with type-replacements and embeddings) on the development set.

these frequent FB-rels can also represent most of the missing relations of nested named entity pairs, we randomly sampled 100 entity pairs in the NL-rel set (Row 1 in Table 5.1) that do not have a Freebase relation and annotated them manually with the 18 FB-rels or *Other* by two annotators. The coverage is 80% by annotator 1 and 92% by annotator 2, with inter-agreement of 71%. The coverage is still satisfactory. Table 5.3 shows the Cohen's kappa coefficient (Equation 5.2) [Smeeton, 1985] for relations that are tagged more than or equal to 5 times by both annotators. Overall kappa value is 0.67, which means the agreement is not accidental. We can see that the two annotators disagree the most on which pair does not belong to the 18 relations, and the */organization/headquarters* relation. The reason is that one annotator prefers to use *-/location/location/contains* instead of */organization/headquarters*.

Cohen's kappa coefficient is used to estimate the agreement between annotators. It is the difference between actual agreement P_o and estimated agreement P_e. If the annotation set is {positive (1), negative (0)},

$$\begin{aligned} k =& \frac{P_o - P_e}{1 - P_e}, \\ P_o =& \frac{(A1 = 1 \quad and \quad A2 = 1)}{all \quad instances} * \frac{(A1 = 0 \quad and \quad A2 = 0)}{all \quad instances}, \\ P_e =& \frac{(A1 = 1)}{all \quad instances} * \frac{(A2 = 1)}{all \quad instances} + \frac{(A1 = 0)}{all \quad instances} * \frac{(A2 = 0)}{all \quad instances} \end{aligned} \tag{5.2}$$

Relations	Kappa
Other	0.24
-/music/artist/album	1.0
-/location/location/contains	0.72
/sports/sports_team/location	0.81
-/location/location/events	0.56
/organization/organization/headquarters /location/mailing_address/citytown	0.51
-/biology/organism	0.92
-/base/cars_refactor/make/model_s	1.0

Table 5.3: The annotation agreement for pairs that have no Freebase relations.

	accuracy (%)
Entity Extraction	99
Definition Sentence	99
FB-rel	96
NL-rel	96

Table 5.4: Automatic annotation accuracy. The first row is the accuracy of entity extraction. The second row shows the accuracy of extracting definition sentences. The third and fourth rows show the accuracy for the FB-rel and NL-rel annotation.

Annotation Accuracy

To estimate the accuracy of our automatic annotation, we sampled 100 nested named entities in the FB-rel set (Row 3 of Table 5.1), and confirmed whether the NL-rel and FB-rel are correct for the super-NE and the sub-NE.

Table 5.4 highlights the results. The first row is the accuracy of entity extraction. The only error is for title *Walton High School (New York City)*, *New York City* is extracted as the super-NE instead of *Walton High School*. The second row shows the accuracy of extracting definition sentences. One sentence out of 100 is incorrect. The third and fourth rows show the accuracy for the FB-rel and NL-rel annotation. FB-rel errors are caused by classifying a frequent relation as *Other*, because of the synonym relations in Freebase. For example, *Camelot Ghana* should have a relation *-/location/location/contains*, but it has */organization/.../headquarters /location/mailing_address/country* instead, which is not in our frequent relation set, thus tagged as *Other*. NL-rel errors are usually caused by parsing mistakes.

5.3 Implicit Relation Classification

We propose classifying the implicit relations with two supervised models: SVM and Neural Network. Before going into details of their functions, we will first describe our feature set.

The entity types play a central role in inferring the relations between two entities. We first extract entity types from Wikipedia definition sentences (by dependency patterns) and from Freebase (by links). However, in reality, entity

#	Feature Description
1	word unigrams in the super-NE
2	token length of the super-NE
3	super-NE's head word
4	super-NE's head word shape
5	super-NE's phrase structure
6.1*	super-NE's Wiki-type
6.2*	sub-NE's Wiki-type
7.1*	super-NE's Freebase type
7.2*	sub-NE's Freebase type
8.1*	bigram frequencies of the Wiki-types and the relation
8.2*	PMI of the Wiki-types and the relation
9+	sub-NE's named entity tag
10.1+	bigram frequencies of the type-replacements and the relation
10.2+	PMI of the type-replacements and the relation
11.1	word embeddings of the super-NE's headword
11.2	entity embeddings of the super-NE
11.3	entity embeddings of the sub-NE

Table 5.5: Features for the implicit relation classification. We use features with * and features with + separately when constructing models in our experiments.

types may not be explicitly expressed in text. To deal with missing types, we replace types with entity head words and named entity tags, such as PERSON and ORGANIZATION. Inspired by the named entity recognition literature we also extract features such as word, word-shape, and word embeddings.

Table 5.5 shows our feature set. In the conversion to machine learning models' instance representation, numerical features, such as PMI scores and token length, are kept in their original form. Categorical features, such as the entity types, word unigrams, and word shapes, are converted into binary features by creating new features for each distinct value. If the category is matching the value is 1, the value is 0 otherwise. Below, we give further explanations for a subset of the features.

Feature 3 and 4 are related to super-NEs' head words. We notice that many compound entities' head words infer the entity types, and the longer an entity, the more likely this is true. For instance, *University of A* usually means the entity is a university. Recall that we already extracted heads of entities when extracting nested named entities from Wikipedia (Section 5.2.1).

The head word's shape feature includes information such as capitalization and existence of numbers. For example, for the entity *Convair XC-99*, the head is *XC-99*. Its shape feature is ALLC-DASH-DIG, where ALLC means that the word is all capitalized and DIG means that the word contains digits.

Feature 5 represents the phrase structure of the super-NE, which is a categorical feature. We embed the entities into sentences "*That is NE,*" e.g., "*That is University of A.*" The Stanford constituent parser will return the phrase structure of the *NE*. For example the phrase structure of *Battle of Gravelines* is "NP $\rightarrow$ NP PP."

Feature 6 includes the super-NE and sub-NE's Wiki-types. These two categorical features are extracted from Wikipedia's definition sentences. We use the dependency path between the subject and the sub-NE again, and assume that the common noun that connects with the subject is the type of the super-NE. Consider again Example (1), its dependency path, *Festival* $\overset{nsubj}{\leftarrow}$ *festival* $\overset{rcmod}{\rightarrow}$ *take* $\overset{prep\_on}{\rightarrow}$ *Isle*, indicates that *festival* is the type of *The Isle of Wight Festival*. We need to use the links in the definition sentences to locate the sub-NE's page and its type. If we can not find the Wiki-type of an entity, it is represented as *null*.

Feature 7 includes Freebase types. Every Freebase entity has several Freebase types. Most of the Wikipedia titles can be mapped to a Freebase entity. We extract entity types in Freebase such as */base/culturalevent/event*, */location/citytown*, and */organization/organization_member*. If we can not find the Freebase type of an entity, it is represented as *null*.

Feature 8 and 10 indicate the correlation between types and implicit relations. Peñas and Ovchinnikova [2012] model the correlation with the probability of a relation given a type. We extend this idea to include the pointwise mutual information (PMI) and bigram frequencies between relations and types (These are two separate features). The bigrams that we consider are: (super-NE's type, relation) and (sub-NE's type, relation). Because in reality, entities' types may not be explicitly mentioned, we also try to replace the Wiki-types with head words or named entity tags. We run the Stanford NER tool on the definition sentences to extract named entity tags [Finkel et al., 2005]. In the

experiments section, we will compare these different types to see the effect on the models' performances.

Feature 11 is a set of features related to embedding. Word embeddings represent words into numerical vectors, which are typically induced by neural language models from a large text. Ideally the more similar two words are, the more similar their vectors will be. One problem of the *one-hot* representation of words, *e.g.*, Feature 1 and 3, is data sparsity. Word embeddings can potentially alleviate the issue. We use word embeddings of super-NEs' head words and the entity embeddings, which are produced by the Word2vec model [Mikolov et al., 2013]. Every embedding represents a set of features with size n, which is the length of an embedding vector.

5.3.1 Supervised models

We compare a support vector machine (SVM) model with a neural network model. We adapt the multi-class SVM tool by [Crammer and Singer, 2002], which uses linear kernel.

For our neural network model, we use the softmax function as the top layer. The predicted probability for the i'th class given the hidden layer $\mathbf{h}$ is:

$$P(y = i|h) = \frac{e^{h^T w_i}}{\sum_k e^{h^T w_k}} \tag{5.3}$$

where h is the hidden layer vector, and w_i is the i_{th} column of the weight matrix W.

For training, we minimize the cross entropy (Equation 5.4) with AdaGrad [Duchi et al., 2011]:

$$Loss = -\sum_{y_i} p(y_i) \log q(y_i). \tag{5.4}$$

where $p(y_i)$ is 1 when y_i is the correct relation of the entity pair, 0 otherwise. $q(y_i)$ is the probability of the relation assigned by the model.

Our neural network model contains only one hidden layer (We also implemented a two layer model which provided minor improvements.) Both SVM and neural network models output the most probable relation as the answer.

5.3.2 Baseline

Peñas and Ovchinnikova [2012] 's model depends solely on entity types to predict relations. We modify their model as a baseline. $P(r_k|e_i, e_j)$, the probability that a relation r_k is implied in an entity pair (e_i, e_j), is equal to:

$$\sum_{t_m \in T_i, t_n \in T_j} P(t_m, t_n|e_i, e_j) \cdot P(r_k|t_m, t_n) \tag{5.5}$$

where t_m is a type in the type set T_i of the entity e_i.

Because every (type, entity) bigram occurs only once in our training data, we set $P(t_m, t_n|e_i, e_j) = 1$ for all type pairs when $t_m \in T_i$ and $t_n \in T_j$.

The probability of a relation given a type pair is calculated based on the counts in the training data:

$$P(r_k|t_m, t_n) = \frac{\#(r_k, t_m, t_n)}{\#(t_m, t_n)} \tag{5.6}$$

If $\#(t_m, t_n) = 0$, we back off to $P(r_k|t_m)$ and $P(r_k|t_n)$:

$$P(r_k|t_m, t_n) = \sqrt{P(r_k|t_m) \cdot P(r_k|t_n)} \tag{5.7}$$

If either $\#(t_m) = 0$ or $\#(t_n) = 0$, we use the other half. If both are zero, we choose the most frequent relation.

5.4 Experiments

Our experiments are divided into two parts: one represents implicit relations in nested named entities with Freebase relations (FB-rel), and the other with natural language relations (NL-rel). Our metric is the weighted average F-score among the relation classes except the *Other* relation[3].

5.4.1 Freebase relation classification

Table 5.6 shows the results of FB-rel prediction on the test set with the baseline (Penãs:2012), SVM, and neural network models. The first four rows represent different feature setting We can see that with nearly perfect entity types the

[3]Weights are assigned by relations' frequency distribution in the test or development set.

#	existence of types and embeddings	SVM	NN
(1)	with Wiki and FB types	94.3	94.4
(2)	(1) + word embedding	93.3	94.2
(3)	with type-replacements	81.7	80.7
(4)	(3) + word embedding	82.2	83.7
(5)	(Penãs:2012) with FB types	79.8	
(6)	(Penãs:2012) with Wiki types	64.8	

Table 5.6: The F-score of FB-rel prediction on the test set. We compare the models with or without Wikipedia or Freebase types; and models with or without word and entity embeddings.

implicit relation extraction is an easy task (Row 1 and 2). For the baseline, using FB types is better than using Wiki types because FB types are less ambiguous. Comparing Row 1 with Row 5, we can see that our feature based model is much better than the baseline. It incorporates features besides types and can predicate relations even when no entity type can be found.

Adding word and entity embeddings does not improve the performance when types are accurate (Row 1 vs. Row 2). Row 3 and 4 show the results without types from Wikipedia or Freebase, instead, we use the super-NE's head and sub-NE's NER tag (type-replacements), i.e., using features 9 and 10 to replace 6 and 8. Under this setting, the embedding feature improves the performance (Row 3 vs. 4)[4].

To better understand the task, we present the precision, recall, and F-score of every class in the last 3 columns of Table 5.2. The results are based on Model (4) in Table 5.6 on the development set. We can see that the performance is not proportional to the training size. *-/symbols/name_source/namesakes* is more frequent than *-/newspaper_circulation_area/newspapers*. But the F-score of the former is much lower than that of the latter. One hypothesis is that the better the head words represent the types, the better the classification performance. For example, the nearby_airports relation gets such a high F-score, because in only 4 out of 228 cases, such nested named entities' extracted head words are neither *airport* nor *aerodrome*. These 4 cases are caused by

[4]The improvement is statistically significant with $P<0.05$, computed using the approximate randomization statistical test.

Gold relation	Confused with
album	Other::12, starring_roles::1
contains	Other::33, nearby_airports::7, tourist_attractions::4, namesakes::2, headquarters::1
nearby_airports	Other::3, contains::1
sports_team/location	Other::2, newspapers::2
location/events	Other::4, contains::2
namesakes	Other::18, album::5, contains::3, organizations_founded::4
compositions	
manufacturer	Other::2
headquarters	contains::17, Other::1, namesakes::1
tourist_attractions	contains::7 , Other::2
people_born_here	Other::15, headquarters::1
starring_roles	Other::2
organizations_founded	Other::5, namesakes::4 , album::1
newspapers	
lower_classifications	Other::2
make/model_s	Other::6
mobile_phone/brand	Other::1
software	Other::3
Other	contains::6, album::4, events::4 namesakes::3 , compositions::3, location::1,headquarters::1, software::1

Table 5.7: Confusion matrix on the development set using the neural network model with type-replacements and embeddings.

head word extraction error.

Table 5.7 shows the confusion matrix of the FB-rel prediction on the development set. Each row shows the correct relation and the false relations assigned by our model, with the corresponding frequency (relations are abbreviated). We can see that relations such as *headquarters*, *tourist_attractions* are mis-classified as *contains*, causing low precision of *contains* and low recalls of *headquarters* and *tourist_attractions*. *model_s* instances are classified as *Others* probably because of the difficulty of inferring types for product named entities.

We also conducted an ablation study to measure the contribution of specific feature sets (Table 5.8). The metric is F-score. The second column represents the study for the model with Wiki and Freebase types. The third column shows the results for the model with type-replacements. From the second column (rm

	Wiki, FB types	type replacements
all features	93.3	80.7
rm 1 (words)	92.8	79.5
rm 2 (NE length)	92.6	80.3
rm 3 and 4 (super-NE head)	92.3	(rm 4) 79.8
rm 5 (phrase structure)	92.5	80.3
rm 6 and 8 (Wiki-type)	92.2	(rm 10) 77.9
rm 7 (FB type)	81.9	N/A
rm 6, 7 and 8	77.3	N/A

Table 5.8: FB-rel prediction F-score on the development set with removing different sets of features. The first row shows results with all features, either with Wiki and FB types or with type-replacements. Other rows show the results of removing one feature set.

#	existence of embeddings	SVM	NN
(1)	with Wiki and FB types	57.5	61.0
(2)	(1) + word embedding	58.1	61.8

Table 5.9: Natural language relation prediction F-score on the test set.

6 and 8 vs. rm 7), we can see that Freebase types are more important than Wiki-types. This may be because Freebase types are very accurate and less ambiguous compared with Wiki-types. Comparing the last row of the second column with the first row of the third column (77.3 vs. 80.7), we can see that adding type-replacements does improve the performance when accurate types are not available. When using type-replacements, the correlation between relations and types becomes the most important feature (rm 10).

5.4.2 Natural language relation classification

Table 5.9 shows the F-scores of NL-rel prediction on the test set with both SVM and neural network models.

The results are much worse than those of FB-rel prediction. This could occur because there is significant overlap between NL-rels. For example, FB-rel */cars_refactor/make/model_s* can be represented as *produce*, *make*, *manufacture*, etc. On the other hand, because of word ambiguity, one word can represent different relations. For example, *develop* can be used to represent the relations between manufacture and airport, or manufacture and software.

Because the automatic NL-rel annotation is too noisy, we recommend using the Freebase relation set to study relations between nested named entities.

5.5 Summary

Previous relation extraction studies concentrate on relations between two entities that do not have an overlap. However there exist an abundance of nested named entities where useful relations exist.

We proposed a new task, implicit relation extraction in nested named entities, to fill the gap in the current relation extraction field. To study the task systematically, we constructed a new dataset automatically using Wikipedia definition sentences and Freebase. We then established a supervised classification model for relation classification. We experimented using features such as Wiki types, entity heads, and word embeddings. Experimental results show that our model is much better than a model which depends solely on types [Peñas and Ovchinnikova, 2012].

Chapter 6

Question Answering with Open Information Extraction

One application of information extraction is supporting natural language understanding, especially in improving other semantic tasks such as machine translation or question answering. After describing several models that improve information extraction, I present our work on using an Open IE model for question answering.

Although there are questions that need inference, such as "*whether a person is guilty given a legal case,*" most current question answering systems focus on answering factoid questions, i.e., questions that can be answered with short phrases about facts. One such example is "*who is the prime minister of Canada in 2016?*" with the answer "*Justin Trudeau.*" Our research also focuses on factoid questions.

Jurafsky and Martin [2000] classify the question answering systems into two categories, information retrieval-based and knowledge-based. The former retrieves answers directly from the webpages with search engines. The latter first builds a semantic structural representation of the knowledge and searches answers from the knowledge base. A knowledge base contains entities and relations between entities. One common format of a knowledge base is a set which contains triples, where one triple contains two arguments and a predicate, i.e., the relation between the two arguments. There are also systems that use both approaches, such as DeepQA from IBM [Ferrucci, 2012]. Our research concentrates on knowledge-based question answering.

The literature further categorizes systems of question answering (QA) from a knowledge base (KB) into two types. One uses semantic parsing [Berant and Liang, 2014, Reddy et al., 2014, Yih et al., 2015], and the other uses information extraction (IE). Semantic parsing systems derive a logical form query from a natural language question and search the knowledge base with this query. One example of the logical form is $\lambda x.road(x) \wedge lead\_to(x, ROME)$, which represents the question "*which road leads to Rome.*" Instead of deriving a logical form, information extraction systems find answers directly, based on the target entity and relation words. Our system belongs to the latter as we do not use logical forms.

There are also at least two types of knowledge bases. One is a curated knowledge base, such as Freebase, which is manually created and very accurate. However, such databases have very low coverage, for example, numerous relations that exist between entities are missing in Freebase. The other is an automatically extracted knowledge base, such as relation triple sets extracted by Open IE systems, which we call IE KB hereafter. Semantic parsing-based systems take the first type as the knowledge base because of their requirement for high accuracy facts. Our system will use both types of knowledge bases. We use both the curated KB, Freebase, and an IE KB, where relations are extracted from a large set of webpages by our Open IE system.

One major challenge for question answering from a KB is the myriad ways in which predicates can be expressed. On the one hand, we need to deal with language variability, for example, acknowledging that the following two questions have the same meaning: "*What character did Natalie Portman play in Star Wars?*" and "*What is the role of Natalie Portman in Star Wars?*" We call this NL-NL paraphrasing, since it requires a mapping between two natural language expressions. On the other hand, we need to bridge the gap between the expression of relations in curated knowledge bases, such as Freebase, and relations conveyed in natural language sentences. We refer to this as NL-KB paraphrasing. For instance, a QA system will require a mapping between the natural language relation *brother* and the Freebase relation "/peo-

ple/person/sibling_s[1]."

There are several levels of paraphrase or textual entailment task: paragraph level, given one paragraph or several sentences, if the other sentence that is not in the paragraph is correct; sentence level, given one sentence, if the other sentence is correct; phrase or word level, given one predicate, i.e., relation, if the other predicate is correct. In the previous example, deciding if the two questions have the same meaning: "*What character did Natalie Portman play in Star Wars?*" and "*What is the role of Natalie Portman in Star Wars?*" it can be done both on the sentence level or the phrase level. On the phrase level, we need to know that <*person, play, character*> has similar meaning to <*person, star as, character*>.

To deal with the NL-KB mapping problem, Berant and Liang [2014], Yao and Van Durme [2014] employed association scores that are learned by machine translation techniques. Alternatively, Fader et al. [2014] used several levels of paraphrases, mapping of questions to templates and relation re-writing. For example, they map question "*who invented papyrus?*" to the query pattern *(?x, rel, arg)*, where *?x* is the slot to be filled, *rel* is the relation, and *arg* is the argument of the relation. Re-write is a phrase level paraphrase, for example, mapping *play as* to *star as*. To calculate the paraphrase scores, i.e., measure the similarity between relations, they use pointwise mutual information (PMI) on different sets of corpora, such as WikiAnswers and their IE KB.

However, there are many other approaches in the paraphrase literature which has not been explored in QA systems. Our research focuses on phrase level paraphrase. We try to find out which paraphrase approach is more suitable for QA by comparing several representative ones, including paraphrase frequency, pointwise mutual information (PMI), and Lin's model DIRT [Lin and Pantel, 2001], which is an advanced version of PMI. Instead of creating several levels of paraphrase, we demonstrate how one layer of phrase level paraphrase can achieve improved performance.

Our overall system is similar to that of [Fader et al., 2014]. One problem

[1]Freebase relation names usually start with type information of the subject entity and end with the relation.

of their system is that their relations in the IE KB are not lemmatized, which adds burden to paraphrase models. For example, the paraphrase process needs to know that "*own*," "*owns*," and "*owned*" represent the same relation. Our IE KB has no such problem.

Another advantage of our KB is that the entities in the KB are linked to the entities in the Freebase. Our system can detect that both "*the United States of America*" and "*U.S.*" represent the same entity as in the Freebase with ID "/m/0248zp." This not only has the potential to improve the paraphrase between NL and FB relations but also improve the recall when searching on the IE KB.

The results on the WebQuestion set [Berant et al., 2013] show that our results improve over those from Fader et al. [2014] and Yao and Van Durme [2014]. Although our results are still inferior to the best question answering system [Yih et al., 2015], our main contribution is providing the new Open IE triple set, and a simplified question answering framework that achieves comparable performance.

6.1 Related Work – Question Answering

A current major research thread in question answering (QA) is to cast the task as a semantic parsing problem, where the objective is to map a natural language question into a formal language, e.g., a database query. This query is then run on a database, and results are returned to the user.

SCISSOR [Ge and Mooney, 2005] was one of the first successful attempts to create a robust semantic parser. It worked by first parsing a question, augmenting the result with semantic information, and then transforming the result into a logical language. However, this process requires a large volume of training supervision, namely "gold standard" annotations of semantically-augmented syntactic trees paired with their logical representations. Its demonstration was limited to GeoQuery [Zelle and Mooney, 1996], which is a database in a very restricted domain.

A more recent approach to achieve robust, open-domain semantic parsing is

that of Berant et al. [2013]. In their approach, they used Lambda Dependency-Based Compositional Semantics (λ-DCS) logical formulas [Liang, 2013] for their meaning representations, which can be converted deterministically into Sparql queries on Freebase.

Along the process, it creates candidate derivations, i.e., trees, specifying derivation rules, with the root node as the output logic form. The training process set weights on features according to an objective function that maximize the log-likelihood of the correct answer. Derivations were constructed recursively based on lexical mapping and composition rules.

The derivations are based on a lexical mapping between the natural language expression and the knowledge base, i.e. NL-KB paraphrase, and a few composition rules, such as intersection of two logical forms: $type.location \bigcap peopleBornHere.Obama$. Their NL-KB paraphrase component consists of two approaches, alignment and bridging.

The alignment is based on the idea that the more entity pairs a natural language relation phrase r, e.g., *brother*, shares with a knowledge predicate p, e.g., /people/person/sibling_s, the more likely they represent the same relation. To extract r, they used REVERB on ClueWeb, a corpus with millions of webpages.

Bridging handles cases where the predicates are expressed weakly or implicitly, such as "*What actors are in Top Gun?*" First they extracted types of the two entities, one answer entity, one target entity *Top Gun*. Then they extracted the implicit predicates according to the type pairs (t1, t2). The more frequent a predicate is with the pairs, the more likely it is the implicit relation.

Yih et al. [2015] developed the current best performing semantic parse-based QA system on the WebQuestion set. Inspired by [Yao and Van Durme, 2014], instead of searching on the whole knowledge base, they defined a query graph which is more closely related to the target entity in the questions. In such restriction, Freebase relations that are not occurred with the target entity will not be the candidates of paraphrase, which prunes the candidates.

Regarding IE QA systems, there are two representatives. Yao and Van Durme

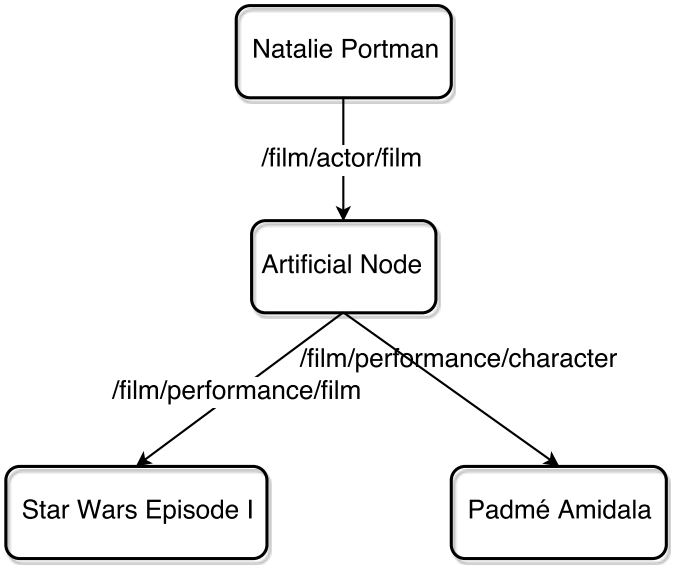


Figure 6.1: An example of a sub-graph for the target entity *Natalie Portman*. It is also an example of compound artificial nodes.

[2014] constructed a graph structure to represent each KB topic. For every question, they first located a subgraph with the target entity in the question, similar to finding related documents in the IR based QA systems. Then they analyzed which entity in the subgraph was most likely to be the answer. Figure 6.1 shows one example sub-graph for the topic entity *Natalie Portman*.

Fader et al. [2014]'s system constructed the KB as a triple database, where each triple consists of two entities and one relation phrase. The relations are from both Freebase and OpenIE extraction. The system exploited several levels of paraphrase.

One is the process of paraphrasing from one question to another question, for example, from "*How does _ affect your body?*" to "*What body system does _ affect?*" They use the corpus of WikiAnswer paraphrase[2] for the step. The corpus contains clusters of questions that are similar. One example cluster contains questions such as "why do we use computers?" and "what do computers replace?"

Another level of paraphrase is *parsing*, which converts natural language questions into a small number of high-precision templates. For example, from

[2]http://wiki.answers.com

"Who/What is NP_{arg}" to (arg, is-a, ?x). The templates are created manually.

Another paraphrase component is query-rewriting, which is a phrase level paraphrase. For example, its paraphrase operator re-writes *children* to *was born to*. As mentioned in the introduction, one problem of their paraphrasing is that it depends on a relation triple set that is not normalized. For example, there are relations *was born to* and *is born to*, which can be easily combined to one relation by lemmatization. The lack of lemmatization will increase the complexity and decrease the performance of paraphrase.

6.2 Our DataSet

To augment our QA system, we have created a new Open IE relation triple dataset that contains sentences and document IDs from which the triples are extracted. The relation triples are extracted from ClueWeb09 by our Open IE system. Each triple contains two arguments, which are entities from Freebase, and one relation phrase. The arguments' Freebase IDs are provided by the FACC1 corpus[3]. We lemmatized the relations and provided the sentences that contain the relation triples. As a result, the dataset contains more than 300 million relation triples[4].

Table 6.1 shows one relation triple example, including the originating parsed sentence. The relation triple is from the 485_{th} sentence of the document *clueweb12-0700tw-51-00204*. The relation word is *director*, which is the 4_{th} word in the sentence. The two arguments *Raul Gonzalez* and *National Council of La Raza* have corresponding Freebase IDs: /m/02rmsx3 and /m/085f3n.

Many NLP tasks can potentially benefit from such data. For example, for the question answering task, there are at least three advantages. One is that the entities are linked to Freebase, which will identify entities that represent one object but with different mentions, for example, mentions in the set {*U.S., United States, United States of America*} represent the same entity. Secondly, the triple is associated with the parsed sentences and the document ID, which

[3]http://lemurproject.org/clueweb09/

[4]We use only half of the ClueWeb09 because of resource limits.

Relation Triple Example: <Raul Gonzalez, director, National Council of La Raza>
<doc>clueweb12-0700tw-51-00204 <relation>485, Raul Gonzalez /m/02rmsx3, National Council of La Raza /m/085f3n, <E1> →(appos) director →(prep_at) <E2> director 4 (ROOT (S (NP (NP (NNP Raul) (NNP Gonzalez)) (, ,) (NP (NP (JJ legislative) (NN director)) (PP (IN at) (NP (NP (DT the) (NNP National) (NNP Council)) (PP (IN of)(NP (NP (NNP La) (NNP Raza)) ...)

Table 6.1: Example of a relation triple extracted from ClueWeb09, with its source sentence and document ID.

can provide better evidence for questions with n-ary relations, such as "*What character did Natalie Portman play in Star Wars?*" Finally, we can provide explanations, i.e., by identifying sentences that are evidence in support of our answers. We believe that this large volume of linked triples may not only improve the mapping between the natural language and Freebase relations, but also improve the recall of question answering, as we can also search answers based on entities' Freebase IDs. For instance, to search "*which movie did Raul Gonzalez direct?*" we can search triples that have an argument *Raul Gonzalez* or /m/02rmsx3.

6.3 System Structure

Figure 6.2 presents our general framework for open question answering. Bear in mind that we search on two KBs: the Freebase and the IE KB. The framework shows how to create a query from a natural language question to search in our KB.

The first component of our system is **query pre-processing**. We use the Stanford CoreNLP tool for entity extraction and sentence parsing [Manning et al., 2014].

We assume each question contains one target entity. For example, in the

question "*What character did Natalie Portman play in Star Wars?*" the target entity is "*Natalie Portman*." Given the dependency parse tree of a question, the priority order of the target entity is: subject > direct object > other entities. Entities with named entity types such as person or organization have higher priority as target entities, compared to those such as numbers and dates, as from our experience most questions are about a person or an organization.

We then extract the question phrase for each question. It can be a single word such as *where* or a multi-word phrase such as *which character*.

To extract target relations, we identify the dependency path between the target entity and the question phrase. The common words on the dependency path are considered to be the target relation words. Consider the example question in Figure 6.2, the dependency path is *character* $\leftarrow$ *play* $\rightarrow$ *Portman*. The relation word is *play*. When there are preposition dependency labels on the path, we create a bigram relation with the verb / noun relation word and the preposition. For instance, for the question "*what country is the grand bahama island in?*" the relation is "*be_in*." "*be*" is filtered out as it is a stop words. Note that we adapt a similar preposition binding approach for the relation triples extracted with our Open IE system. There are cases when there is no common words on the dependency path, one example question is "*what state does selena gomez*?" In such case we extract the words in the question phrase as keywords, such as "*state*." The accuracy of this relation extraction step will be presented in the Experiment section.

The second component of our QA system is **paraphrase recognition**, i.e., the identification of a mapping between relations in the questions and relations in the two knowledge bases. For example, we want to map the target relation "*play*" to relations such as "*star as*" or "*/film/performance/actor*." Further details of our paraphrase models are in Section 6.4.

The third component is **answer retrieval**. Our knowledge base contains triples from both IE KB and Freebase. We retrieve answers based on the target entity, the relation words in the question, the corresponding paraphrase relations, and context words.

The final component is **answer re-ranking**. In the previous answer re-

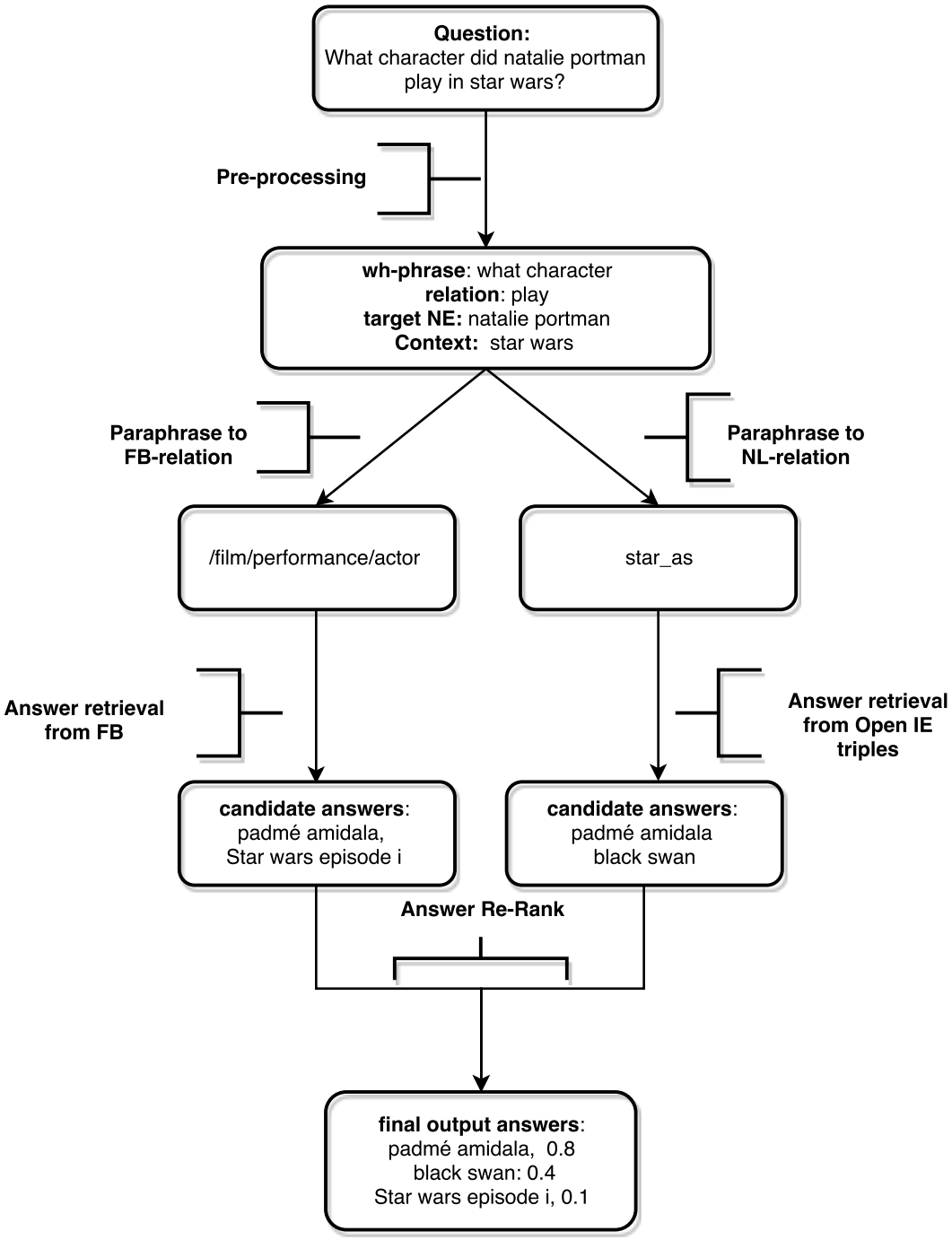


Figure 6.2: Our open question answering system structure.

trieval process, the context of the question, such as the name of the movie for the character, is not used. The type of the retrieved answer and the type of the answer that the question targets are also not considered. In the answer re-ranking step, we extract features related to these information, and then use an SVM-Rank system [Tsochantaridis et al., 2004] to re-rank the candidate answers from the previous step. Every answer will be assigned a score by the re-rank system. The top answers will be returned as the candidate answers. Further details of the model and its features are described in Section 6.6.

6.4 Paraphrase

Substantial research has been devoted to automatic learning of paraphrase from corpora[5]. However, the question answering community has currently chosen only to adapt either a machine translation model or a simple pointwise mutual information (PMI) model for paraphrase. Little has been done to analyze the effect of different paraphrase approaches in question answering systems. To answer the question "will better paraphrase approaches result in better QA performance?" we compare three paraphrase models that we adapt for our question answering system. The fact that our system uses only one layer of paraphrase makes it easier to compare different paraphrase models.

One paraphrase model that we adapt is a frequency based model. Paraphrase rules are learned based on a question answering training set. Given a set of questions, the relation words in the questions are extracted with dependency path rules as mentioned in Section 6.3. Then we retrieve the relation triples that contain both the target entities and the gold standard answers from a KB, which can be FB KB or IE KB. The score of a rule, query_Rel $\rightarrow$ KB_Rel, is the frequency of the entity pairs that the query relation and the KB relation share.

Another paraphrase model is a PMI based model, which is adapted by [Fader et al., 2014] 's QA system. The general function of the PMI between

[5]We concentrate on word/phrase level paraphrase instead of sentence level paraphrase.

two relations x and y is:

$$log\frac{P(x,y)}{P(x)P(y)} \tag{6.1}$$

where $P(x) = count(x)/n$, n is the number of entity pairs in the KB. For paraphrase, $count(relation)$ represents the number of entity pairs associated with the relation, and $count(relation1, relation2)$ represents the number of entity pairs shared by two relations. Comparing with the simple frequency based model, it penalizes relations that are too frequent.

The last paraphrase model we adapt is the highly-cited DIRT algorithm [Lin and Pantel, 2001]. In this model, relations are represented by two vectors. Each vector represents one argument slot. Their similarity score function is defined as follows:

$$\sqrt{sim(v_l^x, v_l^y) * sim(v_r^x, v_r^y)}. \tag{6.2}$$

where v_l^x is a word vector representing the relation x's left argument slot. The value of the vectors is the PMI between the relation and the argument.

We train the PMI and DIRT model on a subset of our Open IE triples. The subset is retrieved with certain constraints to filter out noise in Open IE extraction. One constraint is that, the relation words should appear between the dependency path of two entities. The other is that the maximum path length is 3 (while originally the relation words can be on one side of the entity and the maximum length of the dependency path is 4). These restrictions will increase the accuracy of the triples, and improve the performance of the paraphrase.

Table 6.2 gives one artificial KB which contains both NL-rels and FB-rels. The relations start with slashs are FB relations. There are 8 unique entity pairs (relation directions are not considered). Five pairs of entities have the FB relation */appointees /office_holder*, in which two of them have the NL relation *appoint*. The PMI between */appointees /office_holder* and *appoint* will be $log\frac{2/8}{5/8*3/8}$.

Table 6.3 shows an example of paraphrasing from NL-rels to FB-rels using our real IE KB and FB KB. The NL-rel is *president* and there are several

argument 1	relation	argument 2
benjamin harrison	/appointees /office_holder	william howard taft
alaska house of representatives	/appointees /office_holder	ted stevens
central intelligence agency	/appointees /office_holder	roscoe hillenkoetter
bill clinton	/appointees /office_holder	norman mineta
bill clinton	/appointees /office_holder	pamela harriman
bill clinton	/basic_title	president
bill clinton	/organization_founder	democratic leadership council
benjamin harrison	appoint	william howard taft
bill clinton	appoint	pamela harriman
hillary clinton	appoint	barack obama
roscoe hillenkoetter	director	central intelligence agency
ted stevens	leader	alaska house of representatives
norman mineta	serve	bill clinton

Table 6.2: An artificial KB.

NL-rel: *president*		
FB-rel	PMI	co-occur frequency
/government/us_president/vice_president	4.16	7.0
/government/government_position_held/office_holder	3.21	23.0
/government/government_position_held/appointed_by	3.18	3.0
/military/military_command/military_commander	3.18	3.0

Table 6.3: Paraphrasing from the NL-rel *president* to FB-rels. Second column shows the PMI value between the NL-rel and FB-rel. Third column shows the co-occurrent frequency.

candidates of FB-rel, which are ranked according to their PMI value with the NL-rel. Table 6.4 shows an example of paraphrasing from NL-rels to other NL-rels based on either PMI or DIRT.

6.5 Answer Retrieval

Our answer retrieval process uses a sophisticated form word matching, which is similar to information retrieval. To explain this process, we first give a brief description of the information retrieval architecture as a black box (without details of algorithms).

The input of an information retrieval system is a query with keywords and documents with potential answers for the query. The system searches for relevant documents that contain matches of the keywords, and returns the

NL-rel: *president*		
NL-rel	PMI	co-occur frequency
ceo_of	4.31	370
executive_of	4.31	518
dictator	4.30	410
NL-rel	DIRT	co-occur frequency
head_of	0.05	-
leader	0.03	-
executive	0.03	-

Table 6.4: Paraphrasing from the NL-rel President to other NL-rels. Second column shows PMI or DIRT-score between the NL-rel and others. Third column shows the co-occurrent frequency.

answer in the relevant documents. A keyword can be a MUST, which means the matched document has to contain the keyword. The match could be a more flexible SHOULD, which means the matched document does not have to contain the keyword.

Keywords can be clustered into several fields, with the words in one field of the query only match with the words in that field of the documents. For example, if a query requires to search *language* in the **title** field, *machine learning* in the **article** field, one document with both *machine learning* and *language* terms in the **title**, will only have one match, the term *language*.

In our question answering case, we represent the KB as a set of documents, where each triple is treated as a single document. Our KB contains triples from both IE KB and Freebase. Our IE KB is as described in Section 6.2.

In Freebase, entities are represented as nodes and relations between them are represented as links. One simple method of transforming the graph into triple sets is using the two directly linked entities as arguments of the link relation, such as in [Fader et al., 2014]. However relations between entities can also be represented with several links instead of just one. Our Freebase KB contains relation triples where entity pairs are directly linked by a relation or by a compound relation. Here compound relations are relations with two links. Because we need to map Freebase relations to natural language relations for question answering, too much noise might be added to the mapping process

if we use relations with more links. One example of the compound relation is "/film/actor/film /film/performance/character" in Figure 6.1, which we hereafter abbreviate as "/film /character." The middle node in between is an artificial node, instead of an entity. It is used to represent relations or events with multiple attributes, such as dates and locations. As pointed out by Yao and Van Durme [2014], compound relations are important for the question answering task, adding them will improve the recall of QA systems.

Every document in our KB contains 6 fields: the left argument's surface form (NE1), the right argument's surface form (NE2), the left argument's Freebase ID (FBID1), the right argument's Freebase ID (FBID2), the relation word(s), and the context words. For example, one document has *natalie portman* as the left argument, *padmé amidala* as the right argument, */film /character* as the relation, *m.123* as the FBID1 for the left argument and *m.456* as the FBID2 for the right argument.

The OpenIE triples' context are the sentences containing that triple. The Freebase triples' context are the compound node text for compound relations. For instance, the context of the triple <*natalie portman, /film /character, padmé amidala*> is "*padmé amidala - Star Wars episode ii: attack of the clones - freebase data team - film performance.*"

We use Lucene, an information retrieval framework tool, to store the documents, i.e., triples. Note that context sentences are tokenized and indexed by Lucene, but they are not lemmatized.

Given a question, from question processing and paraphrasing, we create a query which contains a target named entity, a set of relations, and other context words. Our relation set includes both the relations in the question and their paraphrase words, as mentioned in Section 6.4. The Freebase IDs of the target named entity are extracted by the Freebase API[6]. Now the query contains a target named entity, several Freebase ID candidates, a relation set, and a context set. A triple is retrieved if:

- its NE1 or NE2 matches with the target entity, or its FBID1 or FBID2

[6]Google no longer provides the API service after July 2016. However we have the results for all the named entities in the WebQuestion set.

matches with any of the Freebase ID candidates, which is a MUST condition;

- its relation words match with any of the words in the query's relation set, which is a MUST condition;
- its context words match with words in the query's context set, which is a SHOULD condition.

If the NE1 or FBID1 matches the target entity, NE2 is returned as the answer; otherwise NE1 is returned as the answer. The score of a triple is calculated as follows:

$$score(Doc) = \sum_{w_i \in W_Q \cap W_D} Weight(w_i) \tag{6.3}$$

where w_i is a relation word in both the query and the relation instance. We manually set scores to emphasize the original relations over paraphrase terms. The original relations in the question are assigned a weight of 2, while their paraphrase words are assigned with a normalized paraphrase score between $[0, 1]$. The normalization assigns 1 to the best paraphrase relation and 0 to the n_{th}, where n is the maximum number of paraphrases. The scores for other paraphrases are:

$$Weight(w_i) = \frac{score\_para(w_i, rel) - score\_para\_min(w, rel)}{score\_para\_max(w, rel) - score\_para\_min(w, rel)} \tag{6.4}$$

We ignore the score of the entity match as it is a MUST and there is only one target entity. The score of the context match is the number of words shared in the question context and the triple context, which will be used for answer re-ranking. Different relation words can lead to the same answer; we currently sum up the scores of all retrieved triples that lead to one answer.

For explanation purposes, consider again the question "*What character did Natalie Portman play in Star Wars?*" We extract the top three Freebase ID candidates for *natalie portman*: {/m/09l3p, /m/0dncctc, /m/05ngby1}. With the relation *play*, we build a relation set {*play*} ∪ {*/actor, /written_by, /character*} ∪ {*voice, set, quarterback, star, lead*}. The terms {*/actor, /written_by,*

/character} are Freebase relations that have high paraphrase scores with *play*, while the terms {*voice, set, quarterback, star, lead*} are NL relations from the Open IE triples. The context is the word set of the sentence. We then search for the target entity on either the left argument slot or the right, the relation, and the context. The answer "*padme amidala*" is returned based on the relation instances: <padme amidala, /actor, Natalie Portman>, <Natalie Portman, /character, padme amidala> of Freebase; and <Natalie Portman, play, padme amidala>, <Natalie Portman, know, padme amidala>, <Natalie Portman, star, padme amidala> of the Open IE triples.

6.6 Supervised Re-Ranking

Answers retrieved by the previous step are based on word matching. Because of the noise retained in paraphrase and relation extraction, we need to add more information to filter incorrect answers. We train an SVM-Rank model [Tsochantaridis et al., 2004] to re-rank the answers so that correct answers have higher scores than false answers.

Table 6.5 shows the features we experienced for our re-ranking system (Analysis of effects of the features will be presented in the Experiments section.) In the conversion to SVM's instance representation, numerical features such as PMI scores are kept in their original form. Categorical features, such as the types of answers and paraphrase rules, are converted into binary features by creating new features for each distinct value. If the category is matching the value is 1, the value is 0 otherwise. Below, we give further explanations for a subset of the features.

Feature 2 and 3 are categorical features. They add type match constraints with bigrams of (an answer Freebase type, whphrase) or (an answer Freebase type, a relation word). The whphrase and relation words in the question are indications of the correct answer type. For instance, a whphrase *who* indicates the correct answer is more likely to be a PERSON. We use Freebase types as the types of the candidate answers. Freebase types are extracted according the candidate answer's Freebase IDs. For every Freebase ID, there

are relation instances with the relation /type. We extract the right arguments of the instances as the types. There are thousands of types in Freebase, which might cause sparsity issue. We then reduce the types to around 100 clusters, as proposed by [Ling and Weld, 2012]. The type features will consider the types of answers, and diminish those retrieved answers with an inconsistent type.

Feature 4 is a numerical feature which still accounts for type constraints. When the question phrase (whphrase) is a multi-word phrase, it contains the type information. For example, "*which country*" indicates the correct answer should be a *country*. We use our Open IE triples as another source of the candidate answer type information. We observe that types can be considered as a relation between entities. For example, in the triple <*Shakespeare, playwright, England*>, *playwright* is a relation between *Shakespeare* and *England*, and can also be considered as one type of *Shakespeare*. Our hypothesis is that the more often the type word in the question occurs with the candidate answer in the Open IE triple set, the more likely the answer has that type. This hypothesis leads to Feature 4. One example instance is, for the question "*which country invades Poland?*" with the candidate answer *Germany*, the feature value is the number of triples that match <*?, country, Germany*> in the Open IE triple set.

Feature 5, 6, and 7 are the features related to paraphrase rules that lead to the candidate answer. Intuitively, the more accurate the paraphrase rule is, the more likely the answer is correct. We first include the specific rules that lead to the candidate answer as categorical features (Feature 5). We then include the one that has the max score in the format of *max_rule* as categorical features (Feature 6). We also include the value of the score for the *max_rule* as a numerical feature (Feature 7).

Feature 8 is a binary feature, determined by the percentage of context words shared between the question and the retrieved triple (see Chapter 6.5 for the details of relation triples' context). The target entity, the relation words, and whphrase words are excluded from context words. Questions with n-ary relations can benefit from this type of features. For example, for the

ID	Feature description
1 (NameSpace)	Whether the answer is extracted from Open IE KB, Freebase KB, or both.
2 (Types_wh)	Answer Freebase types with whphrase, e.g. */location/country* with *what country*.
3 (Types_rel)	Answer Freebase types and relation in the question, e.g. */location/country* with *invade*.
4 (OpenIE_type)	Frequency of the bigram (answer and the type word in the question) in ClueWeb.
5 (Para)	Paraphrase rules that lead to the answer
6 (ParaMax)	Paraphrase rule with the max score
7 (ParaMaxScore)	The score of the paraphrase rule with the max score
8 (ContextHit)	Context words hitting rate
9 (IRScore)	Score of the answer with the answer retrieval component

Table 6.5: Features for the Supervised System.

question "*What character did Natalie Portman play in Star Wars?*" we check how many words in the set {Star, Wars} are in the candidate answer's triple context. Suppose the two candidate answers are:

1. *padmé amidala*, with context "*padmé amidala - Star Wars episode ii: attack of the clones - freebase data team - film performance*;"

2. *black swan*, with context "*academy award for actress in a leading role - 83rd academy awards - black swan - role: nina sayers - 2010 - nanette - award honor*."

The first answer's *context words hit rate* is 1, as the context has {Star, Wars}. The second has rate of 0. Currently this is a binary feature, if the context words are all matched, the value is 1, 0 otherwise.

In training the SVM re-ranking model, negative instances include the false answers on the top n candidates extracted by answer retrieval from both the Open IE and Freebase, and all the correct answers are as positive instances. This increases the number of positive training instances. In testing, we only re-rank the top n answers, where n is set according to the performance in the development set.

To combine Freebase and the Open IE KB, we have considered two alternatives. One combines the two sets of answers and then re-ranks the combined

answer set. The other is to re-rank the two sets independently, and combine them by assigning weights to different KBs.

6.7 Experiments

We evaluate our question answering system based on the question set provided by Berant et al. [2013]. The questions are generated by the Google Suggest API. Berant et al. [2013] used Amazon Mechanical Turk to create answers for every question. One property of this dataset is that the answers are guaranteed to be found in Freebase.

Our experiments attempt to answer several questions:

- Is our dataset useful for the paraphrase tasks?
- Which popular paraphrase approach is more suitable for question answering?
- Is our system better than other information extraction-based systems?

There are at least two metrics used in the literature for question answering evaluation:

1. Top 1 F_1 score, as used by [Fader et al., 2014]. Every system outputs only one answer. The system's answer is the entity with the highest score (randomly pick one if there is a tie). No answer is produced if the highest score is below a certain threshold. An answer is considered correct if it matches any of the gold standard answers. The precision, recall and F_1 score are calculated globally:

$$\text{Precision} = \frac{\#\text{ questions with correct answers}}{\#\text{ questions with answers}}$$
$$\text{Recall} = \frac{\#\text{ questions with correct answers}}{\#\text{ questions}}$$

2. Average F_1 score. This is used by semantic parsing question answering systems such as [Berant and Liang, 2014, Yih et al., 2015]. For every question, the precision, recall, and F_1 score are computed based on the gold standard answer set. Then, the F_1 scores are averaged across all questions in the test set. This metric is used to reward partially-complete system answers.

In the following experiments, we will compare our system with Fader's with respect to the first metric, and the other systems such as Yih et al. [2015]'s with the second metric.

6.7.1 The Effect of Dataset Size

We demonstrate the effect of different dataset sizes by estimating a paraphrase PMI model from a smaller subset of our data, and then comparing QA systems' performance with these alternative paraphrase sets. We use the recall as the comparative metric. Recall is calculated as the percentage of questions that can be answered by the top 30 candidate answers. To filter out the potential effect of feature choices and supervised models, results are based on the output of the answer retrieval process on Freebase without re-ranking.

For every question we extract the top 100 Freebase (FB) relations as paraphrase of one question relation (NL-rel) (100 is set according to the development set). The paraphrase score is used as weight on the answer retrieval phase. We use the PMI paraphrase model.

Our baseline consists of 800k triples, which is larger than the size of one existing relation triple dataset from Riedel et al. [2013] (200k). The whole paraphrase set has 26 million triples. When using 800k triples, the recall is 10.7%, whereas we obtain a recall of 34.5% when using 26 million triples. We notice that the performance difference is dramatic.

6.7.2 The Effect of Paraphrase

Here we show the corresponding paraphrase effect on the Freebase-based and IE KB-based question answering systems.

As mentioned previously, for the paraphrase between NL-FB relations, we extract the top 100 Freebase relations for one question relation. The paraphrase score is used as a weight on the answer retrieval.

Table 6.6 shows the recall measure on the top 30 answers with alternative paraphrase models based on Freebase. Freq3000 is the case where we use only the 3000 training sentences and the Open IE triple set. Freq3000+PMI is the supervised paraphrase, Freq3000, plus the unsupervised paraphrase with

models	recall on the top 30
Freq3000	63.2%
Freq3000+PMI	64.5%

Table 6.6: Comparing different paraphrase models. Recall on the top 30, based on Freebase.

models	recall on the top 40
Freq3000	40.5%
Freq3000 + PMI	40.8%
Freq3000 + DIRT	40.7%

Table 6.7: Comparing different paraphrase models. Recall on the top 40, based on Open IE KB.

the PMI measure between the FB-NL relations. The results show that the unsupervised paraphrase between FB-NL relations does improve the recall. It is not meaningful to use DIRT here. The reason is that the frequency of every triple is one in Freebase, which makes the frequency of many bigrams (argument, FB-rel) as one. The PMI of (argument, FB-rel), which DIRT depends upon, is not reliable with such low frequencies.

For the paraphrase between question relations and Open IE KB relations (NL-NL), we extract the top $n = 10$ Open IE relations for one question relation (setting n between 6 and 10 does not display much difference on the development set). Table 6.7 shows the top 40 answer recall values, with alternative paraphrase models based on the Open IE KB. There is no obvious difference among the alternatives. The paraphrase between NL-NL relations is more difficult than paraphrase between FB-NL relations. One reason might be that Open IE relation extraction noise is amplified twice for the NL-NL paraphrase.

6.7.3 State-of-the-Art

As mentioned previously, our question answering system is an information extraction based system. To tune the parameters and design the system, we split the training set into 3000 of training sentences and 780 of development sentences. After we tuned the parameters on the development set, we trained the

models	top1_P	top1_R	top1_F_1	avg P	avg R	avg F_1
Fader14	-	-	35	-	-	-
Yao14(FB search API)	-	-	-	51.7	45.8	33.0
OneLayer_FB	44.7	39.5	**41.9**	48.3	45.0	37.2
OneLayer_IE	28.5	26.6	27.5	29.0	24.2	20.4
OneLayer_combine	41.4	40.3	40.8	40.3	45.7	**37.9**

Table 6.8: Results that compare our system with other IE-based question answering systems.

SVM model based on the 3780 training sentences and tested the QA system on the test set. Table 6.8 shows the results of our system comparing with the other two IE based systems. Yao14 [Yao and Van Durme, 2014] 's system uses FB knowledge base solely; while Fader14 [Fader et al., 2014] 's incorporates both the FB and Open IE KB. Our one layer system based on Freebase is much better than the one based on the IE KB due to the noise of information extraction and NL-NL paraphrase. This makes the normal combination method failed, i.e. the two methods mentioned in Section 6.6. Instead, for the combination system we use OneLayer_IE 's results only when OneLayer_FB returns no answers. Our system is better than both the previous IE-based systems. It is better than Fader14 with an absolute F1 gain of 7% although both systems use FB KB and IE KB. Our system based solely on FB KB is already better than Yao14, which also is based on FB KB only. Notice instead of measured manually (as in Fader14), our system is automatically measured on the WebQuestion answer set, which means the performance is under-estimated, as we will show in Section 6.7.5.

We also compare our system with several semantic parsing-based systems: Berant *et al.*'s systems, Berant13 and Berant14; and the Microsoft system, MS15 [Yih et al., 2015], which is a semantic parsing-based system that achieves the current best performance on the WebQuestion set. Table 6.9 shows the results. Our system is the first information extraction based system that performs better than Berant13 on the Freebase data.

models	avg P	avg R	avg F_1
OneLayer_IE	29.0	24.2	20.4
Berant13	48.0	41.3	35.7
OneLayer_FB	48.3	45.0	37.2
OneLayer_combine	40.3	45.7	37.9
Berant14	40.5	46.6	39.9
MS15(FB search API)	49.8	55.7	48.4

Table 6.9: Results that compare our system with semantic parsing-based question answering systems.

6.7.4 Feature Ablation

To measure the contribution of different feature sets, we conduct an ablation study on the development set, i.e., removing one feature set at a time and comparing the results with the results of using the whole feature set. The results are based on Freebase KB. Table 6.10 presents the ablation configurations and results.

It seems the most important features are Freebase types (Feature 2) and information retrieval scores (Feature 9), while the context hitting rate feature (Feature 8) does not help the task at all.

Features removed	avg P	avg R	avg F_1
null	0.506	0.519	0.415
2 (Types_wh)	0.495	0.509	0.405
3 (Types_rel)	0.508	0.516	0.411
4 (OpenIE_type)	0.504	0.517	0.413
5 (Para)	0.499	0.512	0.409
6 (ParaMax), 7 (ParaMaxScore)	0.505	0.517	0.414
8 (ContextHit)	0.507	0.52	0.416
9 (IRScore)	0.489	0.52	0.4

Table 6.10: Feature ablation study results. The first column shows the feature configuration.

6.7.5 Error Analysis

Gold Standard Incompleteness

One problem of using the WebQuestion set as evaluation data is that the gold standard set is incomplete. This is caused both by the incompleteness

of Freebase, and by human annotation mistakes. The results on [Fader et al., 2014] are based on manual judgement of all systems' answers; even if an answer does not match any answer in the gold standard set, it can still be correct. The annotation of *question size* $\times$ *the number of systems to compare* is time consuming. Our results on the previous section are created automatically based on the gold standard set.

To show the effect of incompleteness on the test result, we annotated 400 development set questions, to determine whether the top answers from our OneLayer_IE system are correct. Table 6.11 compares the results on the original answer set and the expanded answer set. With the top1 measure, the absolute difference is 10%. To avoid the manual annotation for every system, future work will expand the answers for all the questions and investigate the effect of the incomplete answer set on the training process.

dataset	top1_P	top1_R	top1_F_1
original	30.9	29.2	30.0
expanded	42.0	39.7	40.8

Table 6.11: The results of the open question answering system on the original development set and the one with expanded answers.

Pre-processing Errors

Recall that we extract one target entities and target relations during preprocessing. For example, for the question "*What character did Natalie Portman play in Star Wars?*", the target entity our system extracted is *Natalie Portman*, and the relation word is *play*. To estimate the performance of preprocessing, we manually annotated 100 sentences in our development set. If the correct target entity or relation word is missing, we tag it as false, i.e., we are using recall as our metric. The recall is 92%.

The main reason for mistakes is because of parsing errors. For example, in the sentence "*who is the seattle seahawks starting quarterback?*" the dependency path is *who* $\leftarrow$ starting $\rightarrow$ *seahawks*, which leads to the extract of *starting* as the relation.

A few misses are because the relation words are not on the dependency path between the question phrase and the target entity. For example, in the sentence "*who is jensen ackles in a relationship with?*" the dependency path is *who* $\rightarrow$ *ackles*. In the future, we will try to use our Open IE system to extract relations instead of this simple dependency pattern-based method.

IE KB Incompleteness

When we look more closely at our systems' errors, we notice that one problem of the IE KB-based systems is that they can not find numbers or common nouns such as *writer* as an answer. This is because of a weakness in the IE triple extraction process, which is designed to extract relations between named entities that are identified by the entity linking systems. Consider these two examples:

Question1 "what kinda music does john mayer sing?"
Gold standard "Rock music".
Our answer "your body is a wonderland", a song by john mayer.

Question2 "what does jennifer lopez do?"
Gold standard "actor"
Our answer "american idol", a TV show where jennifer lopez was a judge.

This naturally creates another thread for future work: expanding the Open IE triple set to include arguments of common nouns or numbers. Then we can find relations between *john mayer* and *Rock music*, which might be *sing*.

6.8 Summary

We have designed and tested a new open question answering (Open QA) framework for question answering over a knowledge base (KB). Our system consists of only one layer of paraphrase, compared to the three layers used in a previous open question answering system [Fader et al., 2014]. However, because

of the more accurate relation triples, and use of linked entities from IE KB to Freebase, our system achieves a 7 % absolute gain in F_1 score over the previous Open QA system.

Chapter 7

Conclusions and Future Work

7.1 Summary

In contrast to traditional relation extraction, which extracts a specific set of relations, open information extraction (Open IE) attempts to extract all relations. The task uses words in the sentences to express relations instead of categorizing them as in the traditional relation extraction. This dissertation introduced several Open IE systems and their application to question answering (QA).

Instead of using only features such as part of speech and function words as in ReVerb, our Open IE systems are based on dependency trees. Instead of solely depending on patterns or rules, we trained SVM models to classify true and false relations from relation candidates that are extracted by patterns. SVM tree kernels have been proved to be effective for extracting features implicitly from tree structures. Chapter 3 presented our first tree kernel-based Open IE system that utilized dependency parse trees. Chapter 4 introduced our second tree kernel-based Open IE system that incorporated word information into the tree kernel. We represented words as word embeddings learned from large unannotated corpus.

One limitation of previous Open IE systems is that they did not extract relations from noun phrase structures such as *Shakespeare's books*, *Google Images*. In Chapter 5 we then investigated the implicit relations in nested named entities, one type of noun phrase structures. We automatically extracted thousands of training data from Wikipedia and Freebase. We then explored differ-

ent features for the task and built a supervised machine for extracting implicit relations.

To show the effectiveness of our Open IE system, we built an QA system based on the relation triples which are extracted by our Open IE system. The relation triples are used both for paraphrase and as a knowledge base. Our system's performance beats another Open IE-based QA system.

7.2 The Impact of this Work

Our work on Open IE is closely related to and inspired by parsing techniques. Currently parsing systems are relatively slow comparing with word segmentation, part of speech tagging, or named entity recognition systems. However, the author believes that there are structures in natural languages. To build natural language understanding systems, structure models will be superior to sequential models eventually. Our two Open IE systems are evidences for the theory. Our first model, which builds on dependency parse information, is statistically significantly better than ReVerb. Our second model, which was the first that leverages both lexical and syntactical features for the Open IE task, achieved even better results, which is similar to the relation between lexicalized parsing and unlexicalized parsing techniques.

Our work also explores the question "*what is a relation?*" Originally it was defined as verbs in sentences in [Banko et al., 2007, Fader et al., 2011]. There are obvious evidence that this is not sufficient. For example, in "*Agfa Corp., a unit of Bayer AG.*" *unit* is the relation between *Agfa Corp.* and *Bayer AG*. Our two Open IE models extend the relation range to common words: nouns, verbs, and adjectives.

We also notice that there are relations implied in nested named entities. For instance, there is a *write*, or *author*, relation in *Shakespeare's books*; there is an *own* relation in *Google Images*. In Chapter 5, we proposed a method to automatically acquire thousands of training instances for implicit relations in nested named entities and built a model to show the feasibility of the task. This dissertation advocates further exploration on this direction.

7.3 Future Work

7.3.1 Open Information Extraction

From the experience of working on the QA system, we realized that the current Open IE systems still have room for improvement, especially on noisy data where there are abundance of spelling and grammatical errors.

A possible direction for improvement over current Open IE systems could be creating a new word embedding representation set. For that set we can use our relation triples, which were used in Chapter 6 for paraphrasing. This semantic structure-based word embedding might be more suitable for Open IE than the one trained on plain text. When training word embedding models on plain text, there is considerably more noise for context words and important context terms can be missing. For example, in the sentence "European scientists discovered a new star with the ESO telescope," with the sequential structure, the context words for *discover* within distance two will be {European, scientist, new, star}; with the relation structure, the context set becomes {scientist, star, telescope}. Intuitively we will prefer the second set as *telescope* is more closely related to *discover* than *new*.

Another direction we could take is to explore different machine learning models. One possibility would be leveraging deep learning models, which have shown robust performance on noisy data, such as web pages, for tasks such as syntactic parsing. [Miwa and Bansal, 2016] proposed a long short-term memory recurrent neural network on dependency path for joint traditional relation extraction and entity extraction. Their system achieved comparative performance to relation extraction. An advantage of the neural network models over our lexicalized tree kernel model is that they can be coupled with word embeddings tightly and make updating word embeddings according to our training data relatively easier than the SVM model.

A potential disadvantage of using deep learning models is the lack of training data for Open IE. We can create data automatically from the Propbank corpus, an semantic role labeling corpus that contains annotated arguments for different verbs [Palmer et al., 2005]. Below, we give an example sentence an-

notation from Propbank for the verb instances "donate" (1) and "broadcast" (2).

(1) [$_{AM-CAU}$ Due to the earthquake in San Francisco], Nissan is [$_{V}$ donating] [$_{A1}$ its commercial air time] [$_{AM-PRP}$ [$_{A0}$*PRO*-1] to broadcast American Red Cross Emergency Relief messages].

(2) Due to the earthquake in San Francisco, Nissan is donating its commercial air time [$_{A0}$ *PRO*-1] to [$_{V}$ broadcast] [$_{A1}$ American Red Cross Emergency Relief messages].

Here V represents the predicate. $AM-CAU$, $A0$, $A1$, and $AM-PRP$ are different classes of arguments defined by the Propbank annotators. *PRO*-1 is not a word in the sentence but a link corresponding to NP-SBJ-1 node in the constituent parse tree, i.e. *Nissan*.

For Open IE, we propose the following tagging format:

(3) Due to the earthquake in San Francisco, [$_{E1}$ Nissan] is [$_{R}$ donating] [$_{E2}$ its commercial air time] *PRO*-1 to broadcast American Red Cross Emergency Relief messages.

(4) Due to the earthquake in San Francisco, [$_{E1}$ Nissan] is donating its commercial air time *PRO*-1 to [$_{R}$ broadcast] [$_{E2}$ American Red Cross Emergency Relief messages].

Here R represents relation word(s), $E1$ and $E2$ are two entities.

We could train a supervised model mapping from the Propbank annotation to the Open IE annotation with our annotation on Treebank set as Treebank shares the same articles with Propbank. A simple baseline is hand crafting patterns that account for a large portion of the mappings between semantic role labeling and Open IE. An example could be, if a verb has argument one and argument two in the Propbank, then it is very likely that this is a triple.

In terms of nested named relation extraction, instead of continuing to use wikipedia data, future work should extract more evidence of nested named entities from news articles. This can be achieved by models such as [Lu and

Roth, 2015] , which claim to be able to extract nested named entities.

We could also attempt to utilize a character based deep learning model for the implicit relation extraction. Currently the model is at the word level and manually engineered features. However, according to the result on Chapter 5, we can see that there is still room for improvement when the entity type feature is not available. Character level models have the potential to capture the implicit types based on the characters.

Finally, we could adapt our Open IE systems to other domains and languages. One advantage of extracting training data from Propbank is that the Propbank dataset includes annotation on different languages. With language dependent preprocessing such as word segmentation, named entity extraction, parsing, our SVM models should be able to adapt to different languages easier than pattern based Open IE models.

Our implicit relation extraction might benefit the IE in medical domain, the text of which contains abundance of nested named entities. IE in medical domain can help doctors diagnose diseases from symptoms. One example of implicit relation in medical domain is *murine PU.1 promoter*, which implies a *Protein-Component* relation between the two entities, *murine PU.1 promoter* and *PU.1*.

7.3.2 Beyond Open IE

In Chapter 6, we showed how our Open IE system can be used in the question answering task. There are a number of other important NLP problems where Open IE could have an impact. One such problem is information retrieval. The most prevalent IR systems depend on bag of words representation of documents. Open IE systems can extract structured information from documents, which will potentially eliminate less related information.

Bibliography

Alan Akbik. Wanderlust : Extracting semantic relations from natural language text using dependency grammar patterns. In *WWW 2009 Workshop on Semantic Search*, volume 137, pages 279–290, 2009.

Michele Banko, Michael J Cafarella, Stephen Soderl, Matt Broadhead, and Oren Etzioni. Open information extraction from the web. In *In IJCAI*, pages 2670–2676, 2007.

Jonathan Berant and Percy Liang. Semantic parsing via paraphrasing. In *Association for Computational Linguistics (ACL)*, 2014.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. Semantic parsing on freebase from question-answer pairs. In *Proceedings of EMNLP*, 2013.

Peter F. Brown, Peter V. deSouza, Robert L. Mercer, Vincent J. Della Pietra, and Jenifer C. Lai. Class-based n-gram models of natural language. *Comput. Linguist.*, 18(4):467–479, December 1992. ISSN 0891-2017.

Razvan C. Bunescu and Raymond J. Mooney. A shortest path dependency kernel for relation extraction. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, pages 724–731, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics. doi: http://dx.doi.org/10.3115/1220575.1220666. URL `http://dx.doi.org/10.3115/1220575.1220666`.

Yee Seng Chan and Dan Roth. Exploiting syntactico-semantic structures for relation extraction. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 551–560, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics. ISBN 978-1-932432-87-9.

Janara Christensen, Mausam, Stephen Soderland, and Oren Etzioni. An analysis of open information extraction based on semantic role labeling. In *Proceedings of the sixth international conference on Knowledge capture*, K-CAP '11, pages 113–120, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0396-5. doi: 10.1145/1999676.1999697. URL `http://doi.acm.org/10.1145/1999676.1999697`.

Michael Collins and Nigel Duffy. Convolution kernels for natural language. In *Advances in Neural Information Processing Systems 14*, pages 625–632. MIT Press, 2001.

Michael Collins and Nigel Duffy. New ranking algorithms for parsing and tagging: kernels over discrete structures, and the voted perceptron. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 263–270, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics.

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

Linguistic Data Consortium. Ace (automatic content extraction) english annotation guidelines for relations (version 6.2). 2008.

Koby Crammer and Yoram Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *J. Mach. Learn. Res.*, 2:265–292, March 2002. ISSN 1532-4435.

Danilo Croce, Alessandro Moschitti, and Roberto Basili. Structured lexical similarity via convolution kernels on dependency trees. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 1034–1046, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics. ISBN 978-1-937284-11-4.

George R. Doddington, Alexis Mitchell, Mark A. Przybocki, Lance A. Ramshaw, Stephanie Strassel, and Ralph M. Weischedel. The automatic content extraction (ace) program - tasks, data, and evaluation. In *LREC*. European Language Resources Association, 2004.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.*, 12: 2121–2159, July 2011. ISSN 1532-4435.

Anthony Fader, Stephen Soderland, and Oren Etzioni. Identifying relations for open information extraction. In *EMNLP (2011)*, 2011.

Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. Open question answering over curated and extracted knowledge bases. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, pages 1156–1165, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2956-9.

D. A. Ferrucci. Introduction to "this is watson". *IBM J. Res. Dev.*, 56(3): 235–249, May 2012. ISSN 0018-8646. doi: 10.1147/JRD.2012.2184356.

Jenny Rose Finkel and Christopher D. Manning. Nested named entity recognition. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1 - Volume 1*, EMNLP '09, pages 141–150, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics. ISBN 978-1-932432-59-6.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 363–370, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics. doi: 10.3115/1219840.1219885.

Ruifang Ge and Raymond J. Mooney. A statistical semantic parser that integrates syntax and semantics. In *Proceedings of the Ninth Conference on Computational Natural Language Learning*, CONLL '05, pages 9–16, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics. URL http://dl.acm.org/citation.cfm?id=1706543.1706546.

Roxana Girju, Dan Moldovan, Marta Tatu, and Daniel Antohe. On the semantics of noun compounds. *Comput. Speech Lang.*, 19(4):479–496, October 2005. ISSN 0885-2308.

G. Golub and W. Kahan. Calculating the singular values and pseudo-inverse of a matrix. *Journal of the Society for Industrial and Applied Mathematics: Series B, Numerical Analysis*, page 205224, 1965.

Dirk Hovy, Stephen Tratz, and Eduard H. Hovy. What's in a preposition? dimensions of sense disambiguation for an interesting word class. In *COLING 2010, 23rd International Conference on Computational Linguistics, Posters Volume, 23-27 August 2010, Beijing, China*, pages 454–462, 2010.

Daniel Jurafsky and James H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition*. Prentice Hall, 2000.

Dan Klein and Christopher D Manning. Fast exact inference with a factored model for natural language parsing. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 3–10. MIT Press, 2003.

Pavel Kuksa, Yanjun Qi, Bing Bai, Ronan Collobert, Jason Weston, Vladimir Pavlovic, and Xia Ning. Semi-supervised abstraction-augmented string kernel for multi-level bio-relation extraction. In *Proceedings of the 2010 European conference on Machine learning and knowledge discovery in databases: Part II*, ECML PKDD'10, pages 128–144, Berlin, Heidelberg, 2010. Springer-Verlag. ISBN 3-642-15882-X, 978-3-642-15882-7.

P. Liang. Semi-supervised learning for natural language. Master's thesis, Massachusetts Institute of Technology, 2005.

Percy Liang. Lambda dependency-based compositional semantics. *CoRR*, abs/1309.4408, 2013. URL http://arxiv.org/abs/1309.4408.

Dekang Lin and Patrick Pantel. DIRT @SBT@discovery of inference rules from text. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '01, pages 323–328, New York, NY, USA, 2001. ACM. ISBN 1-58113-391-X. doi: 10.1145/502512.502559.

Xiao Ling and Daniel S. Weld. Fine-grained entity recognition. In *In Proc. of the 26th AAAI Conference on Artificial Intelligence*, 2012.

Wei Lu and Dan Roth. Joint mention extraction and classification with mention hypergraphs. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 857–867, Lisbon, Portugal, September 2015. Association for Computational Linguistics. URL http://aclweb.org/anthology/D15-1102.

Bill MacCartney, Trond Grenager, Marie-Catherine De Marneffe, Daniel Cer, and Christopher D. Manning. Learning to recognize features of valid textual entailments. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, HLT-NAACL '06, pages 41–48. Association for Computational Linguistics, 2006.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, 2014.

Marie-Catherine De Marneffe and Christopher Manning. The stanford typed dependencies representation. In *Coling 2008: Proceedings of the Workshop on Cross-Framework and Cross-Domain Parser Evaluation*, CrossParser '08, pages 1–8, Stroudsburg, PA, USA, 2008. Association for Computational Linguistics. ISBN 978-1-905593-50-7.

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

Marie-Catherine De Marneffe, Bill MacCartney, and Christopher D. Manning. Generating typed dependency parses from phrase structure parses. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC-2006)*, Genoa, Italy, May 2006b. European Language Resources Association (ELRA). ACL Anthology Identifier: L06-1260.

Mausam, Michael Schmitz, Robert Bart, Stephen Soderland, and Oren Etzioni. Open language learning for information extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, page 523534. Association for Computational Linguistics, 2012.

Filipe Mesquita, Ying Xu, Denilson Barbosa, Grzegorz Kondrak, Aditya Bhargava, and Mirko Bronzi. The effectiveness of traditional and open relation extraction for the slot filling task at TAC 2011. In *Text Analysis Conference (TAC)*, 2011.

Filipe Mesquita, Jordan Schmidek, and Denilson Barbosa. Effectiveness and efficiency of open relation extraction. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 447–457. Association for Computational Linguistics, 2013.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In C.j.c. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. 2013.

Makoto Miwa and Mohit Bansal. End-to-end relation extraction using lstms on sequences and tree structures. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1105–1116, 2016.

Alessandro Moschitti. *Automatic text categorization: from information retrieval to support vector learning*. Aracne, 2005. ISBN 8854802921.

Alessandro Moschitti. Efficient convolution kernels for dependency and constituent syntactic trees. In *Proceedings of the 17th European conference on Machine Learning*, ECML'06, pages 318–329, Berlin, Heidelberg, 2006. Springer-Verlag. ISBN 3-540-45375-X, 978-3-540-45375-8.

Truc-Vien T. Nguyen, Alessandro Moschitti, and Giuseppe Riccardi. Convolution kernels on constituent, dependency and sequential structures for relation extraction. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 3 - Volume 3*, EMNLP '09, pages 1378–1387, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics. ISBN 978-1-932432-63-3. URL `http://dl.acm.org/citation.cfm?id=1699648.1699684`.

Sebastian Padó. *User's guide to* `sigf`*: Significance testing by approximate randomisation*, 2006.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. The proposition bank: An annotated corpus of semantic roles. *Comput. Linguist.*, 31(1):71–106, March 2005. ISSN 0891-2017.

Anselmo Peñas and Ekaterina Ovchinnikova. Unsupervised acquisition of axioms to paraphrase noun compounds and genitives. In *Proceedings of the 13th International Conference on Computational Linguistics and Intelligent Text Processing - Volume Part I*, CICLing'12, pages 388–401, Berlin, Heidelberg, 2012. Springer-Verlag. ISBN 978-3-642-28603-2.

Barbara Plank and Alessandro Moschitti. Embedding semantic similarity in tree kernels for domain adaptation of relation extraction. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1498–1507, Sofia, Bulgaria, August 2013. Association for Computational Linguistics.

Siva Reddy, Mirella Lapata, and Mark Steedman. Large-scale semantic parsing without question-answer pairs. *Transactions of the Association for Computational Linguistics*, 2:377–392, 2014.

Sebastian Riedel, Limin Yao, Benjamin M. Marlin, and Andrew McCallum. Relation extraction with matrix factorization and universal schemas. In *Joint Human Language Technology Conference/Annual Meeting of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL '13)*, June 2013.

Xin Rong. word2vec parameter learning explained. *CoRR*, abs/1411.2738, 2014.

Barbara Rosario and Marti Hearst. Classifying the semantic relations in noun compounds via a domain-specific lexical hierarchy. In *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing (EMNLP-01*, pages 82–90, 2001.

Nigel C. Smeeton. Early history of the kappa statistic. *Biometrics*, 41(3): 795–795, 1985.

Richard Socher, Brody Huval, Christopher D. Manning, and Andrew Y. Ng. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2012.

Richard Socher, John Bauer, Christopher D. Manning, and Andrew Y. Ng. Parsing with compositional vector grammars. In *ACL*. The Association for Computer Linguistics, 2013.

Vivek Srikumar and Dan Roth. Modeling semantic relations expressed by prepositions. *TACL*, 1:231–242, 2013.

Shashank Srivastava, Dirk Hovy, and Eduard Hovy. A walk-based semantically enriched tree kernel over distributed word representations. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1411–1416, Seattle, Washington, USA, October 2013. Association for Computational Linguistics.

Ioannis Tsochantaridis, Thomas Hofmann, Thorsten Joachims, and Yasemin Altun. Support vector machine learning for interdependent and structured output spaces. In *Proceedings of the Twenty-first International Conference on Machine Learning*, ICML '04, pages 104–, New York, NY, USA, 2004. ACM. ISBN 1-58113-838-5. doi: 10.1145/1015330.1015341.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 384–394, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

Fei Wu and Daniel S. Weld. Open information extraction using wikipedia. In *ACL-2010*, 2010.

Feiyu Xu, Hans Uszkoreit, and Hong Li. A seed-driven bottom-up machine learning framework for extracting relations of various complexity. In *Proceedings of ACL 2007, 45th Annual Meeting of the Association for Computational Linguistics*, pages 584–591, Prague, Czech Republic, 6 2007.

Ying Xu, Mi-Young Kim, Kevin Quinn, Randy Goebel, and Denilson Barbosa. Open information extraction with tree kernels. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 868–877, Atlanta, Georgia, June 2013. Association for Computational Linguistics.

Ying Xu, Christoph Ringlstetter, Mi-young Kim, Grzegorz Kondrak, Randy Goebel, and Yusuke Miyao. A lexicalized tree kernel for open information extraction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 279–284, Beijing, China, July 2015. Association for Computational Linguistics.

Xuchen Yao and Benjamin Van Durme. Information extraction over structured data: Question answering with freebase. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 956–966. Association for Computational Linguistics, 2014. URL `http://aclweb.org/anthology/P14-1090`.

Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Proceedings of the Joint Conference of the 53rd Annual Meeting of the ACL and the 7th International Joint Conference on Natural Language Processing of the AFNLP*. ACL Association for Computational Linguistics, July 2015.

John M. Zelle and Raymond J. Mooney. Learning to parse database queries using inductive logic programming. In *Proceedings of the Association for the Advancement of Artificial Intelligence*, pages 1050–1055, 1996.

Min Zhang, Jie Zhang, Jian Su, and Guodong Zhou. A composite kernel to extract relations between entities with both flat and structured features. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, ACL-44, pages 825–832, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics. doi: 10.3115/1220175.1220279. URL `http://dx.doi.org/10.3115/1220175.1220279`.

Guo-Dong Zhou and Qiao-Ming Zhu. Kernel-based semantic relation detection and classification via enriched parse tree structure. *J. Comput. Sci. Technol.*, 26(1):45–56, January 2011. ISSN 1000-9000.

GuoDong Zhou, Jian Su, Jie Zhang, and Min Zhang. Exploring various knowledge in relation extraction. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 427–434, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics.

Guodong Zhou, Min Zhang, Donghong Ji, and Qiaoming Zhu. Tree kernel-based relation extraction with context-sensitive structured parse tree information. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 728–736, 2007.