High-quality Knowledge Acquisition of Predicate-argument Structures for Syntactic and Semantic Analysis

Gongye Jin

March 2016

Abstract

In computer science, people have been trying hard to march toward one of the very ultimate goals which expects computers can truly understand human's language. This task is named as text understanding. During the past decades, great effort has been made to develop various kinds of techniques to achieve text understanding which can further provide us the space to discover more humanized ways for human-computer interaction.

To complete this ultimate goal, it may involve multiple different domains such as cognitive science, linguistic and even psychology. In Natural Language Processing (NLP) that is a field of computer science and artificial intelligence, for computer to understand text in human language, the priority is to discover and clarify the internal relations of the words within a sentence. That is to say, computers should have the ability to automatically analyze the input text to discover the interaction among all the words, in order to further comprehend all the contents described in the text. Many NLP techniques such as syntactic parsing and semantic analysis is needed to achieve this kind of automatic analysis.

One can build such a system either based on a massive amount of manually defined rules or utilize machine learning approaches which require a certain amount of human-annotated training data to learn a statistic model. Even though simple cases can be solved by sets of rules, but for complicated problems, even making great effort on defining huge sets of rules cannot achieve a good performance. Machine learning approaches are always preferred in these analyses. One big drawback of machine learning approaches is the requirement for human-annotated data that is always time-consuming to create. Thus machine learning-based systems are always limited in performance by the size of training data. In addition, according to the characteristics of different languages, some languages hit a bottleneck because of its hard-to-analyze properties, such as omission, word order

and lack of inflection. Take Chinese as an example, because there is no inflection that is a common phenomenon in English, it is always hard to infer the tense, case and voice of the predicate. Even Chinese native speakers can be sometimes confused about this type of case, unless utilizing the prior knowledge. Therefore, in this thesis, it is considered to be a crucial approach to acquire large-scale knowledge to compensate the drawbacks in NLP.

Such knowledge can also be constructed manually. However, besides its difficulty for compilation, manual effort is always not enough to solve the issues such as knowledge coverage, updating problem etc. Thanks to data explosion, unprocessed data that potentially contain a large amount of useful information can be relatively easier to acquire, it is thus promising to acquire necessary information from these raw texts using automatic approaches. Automatic knowledge acquisition is always dependent on the abovementioned automatic analysis such as syntactic parsing and semantic analysis. Without considering the automatic analyzing errors, the knowledge will be extremely noisy and may cause bad effects when we apply this kind of knowledge to other tasks in NLP. In this thesis, we solve the abovementioned problems by proposing a framework of creating high-quality knowledge from unlabeled raw text that contains less noise, and apply the knowledge to different NLP tasks.

In Chapter 1, we give some detail introduction about the knowledge acquisition approach from raw text and discuss some major problems in this approach. Then the proposed framework for knowledge acquisition will be introduced.

In Chapter 2, we describe an approach which deals with the erroneous automatic analysis. This approach automatically selects high-quality syntactic parses in order to suppress the noisy problem in knowledge construction.

In Chapter 3, we introduce a type of knowledge called *Case Frames* which are composed by syntactic parses.

In Chapter 4, we use case frames to support syntactic parsing.

In Chapter 5, we apply case frames to *Semantic Role Labeling* which is considered to be a semantic level analysis.

In Chapter 6, we propose an approach to automatically select high-quality semantic roles from automatic analyses in order to construct more representative knowledge.

Acknowledgments

I would like to express my sincere appreciation to Associate Professor Daisuke Kawahara and Professor Sadao Kurohashi for their supervision of this thesis and my whole research. For six years, they continuously encouraged and inspired me throughout this research and guided me in the right direction.

I also would like to express my gratitude to Professor Tatsuya Kawahara for his constructive suggestions during my research. I would like to thank Associate Professor Shinsuke Mori who kindly offered my many inspiring advices for my research.

I am grateful to Dr. Tomohide Shibata who gave my many help during these years.

I am also grateful to Dr. Mo Shen who kindly offered me a lot of help with dependency parsing technique.

I would like to thank Dr. Chenhui Chu who gave me a lot of machine translation related information.

I would like to thank Mr. Raj Dabre who gave me many inspiring advices for my research.

I would like to thank Mr. Arseny Tolmachev who kindly helped me with coding problems.

I am grateful to the previous and current members from Kurohashi-Kawahara laboratory. Especially, I would like to thank Mr. Kenichi Otomo who kindly gave me all kinds of support when I was a new comer of this laboratory. I would like to thank my previous tutor Mr. Hiroshi Manabe along with Mr. Yuki Kurokawa.

Finally, I would like to thank my family and friends for their continuous supports and encouragements during and before this work.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Over the past decade, along with the development of computer science, the challenge to enable computers to understand human languages has received a lot of attention. Although it is quite an ambitious goal, the very first step that should be taken in order to achieve this goal is to identify various types of relations within each sentence in human languages, especially the relations between predicates and their grammatically or semantically related arguments. Due to the fact that the training data for this kind of automatic approaches is not always enough to construct a robust machine learning-based system, in order to precisely capture these relations, a knowledge base with a wide coverage is indispensable. In this chapter, we give a brief introduction of knowledge acquisition approaches and related Natural Language Processing (NLP) tasks which are essential prerequisites.

1.1 Fundamantal NLP Tasks

The automatic analysis of text requires large amount of fundamental NLP processing. The various NLP processing steps are as follows:

Morphological Analysis: This task focuses on the identification of a given language's morphemes, that are the smallest grammatical units in a language. It also identifies other linguistic units such as root words, affixes, part-of-speech, intonations and stresses. A morpheme may function as a content word when it has a meaning of

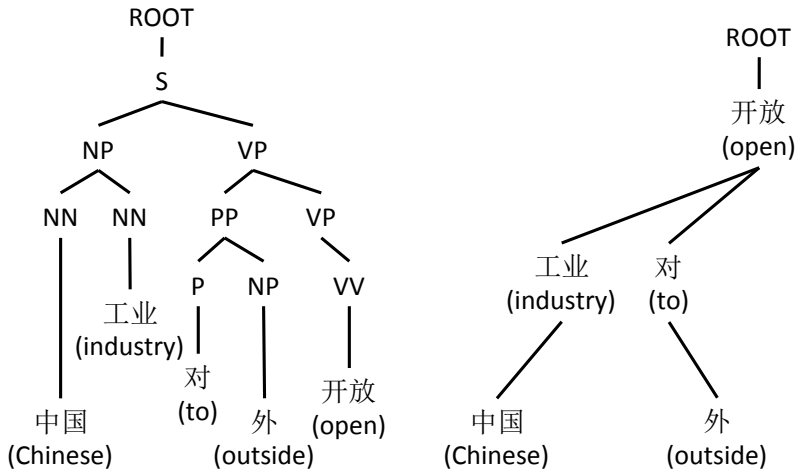

Figure 1.1: Examples of syntactic parsing

its own (e.g., the morpheme "apple"). It may also play a role as an affix/function word when it has a grammatical function (e.g., the *-s* in *"apples"* to specify that it is plural). For many Asian languages, this approach is mostly used to segment the whole sentence into morphemes and assign a part-of-speech tag to each morpheme.

Syntactic Parsing: This task discovers the formal rules of how words are composed to form a whole sentence following language specific grammars. Usually, this approach transforms a sentence into tree structures, where a sentence is structured into its constituents. This kind of structure can show a general picture for sentence comprehension. Syntactic parsing can have another type of tree representation using dependency grammar, where words are connected to each other by directed links, showing that one word is grammatically affiliated to another one. Figure 1.1 shows two examples to give the intuition of syntactic parsing.

The first type is usually called constituent parsing. This type of syntactic parsing divides a sentence into constituents that represent a rough general grammatical structure. More refined structures can be further derived from each constituent according to certain linguistic grammars. The second type is known as dependency parsing. In a dependency parsing tree, each directed edge links a word pair which consists of a head and its modifier. This kind of dependency structure generally in-

dicates the grammatical importance, i.e., a head contains more information than its affiliated modifier in most cases. Also in some cases, the dependency link contains a syntactic label to indicate what kind of syntactic role a modifier node is playing (e.g., the modifier is an *object* of the head). Thus, this kind of structure is always convenient for the purpose of information extraction from a sentence.

Semantic Role Labeling: This task is mainly used to clarify deeper-level semantic relations (e.g., [*who*] does [*what kind of*] thing to [*whom*] in [*what time*]) within the sentence. For text understanding, it is crucial to find out semantic-level information such as which word indicates the status of an event, which person/thing is involved in this event or which subject acts/receives a certain action. This task is always based on the previous-level approaches such as morphological analysis and syntactic parsing. As a result, the performance of semantic role labeling depends much on the performance of previous processes. Similar to syntactic parsing, there are also two methodologies of semantic role labeling. The first one is consitituent-based which assigns semantic roles to the constituents or phrases in sentences. The second one transforms the sentence into a dependency structure, where directed edges are tagged with semantic labels in order to indicate the semantic roles of each argument.

To show the multi-level process for text understanding, we take the following Chinese sentence as an example:

中国 (Chinese) 工业(industry) 对(to) 外 (outside) 开放 (open)

which means “The Chinese industry is internationalized”. Roughly, this is translated as “The Chinese industry is open to the outside (world)”. After the morphological analysis which determines the word boundaries, syntactic parsing is applied, in most cases, to automatically recognize the grammatical relations, such as “工业(industry)” being a subject of “开放 (open)”. In semantic analysis, word sense disambiguation of the verb “开放 (open)” is applied. Also the semantic relations are discovered (e.g., “工业(industry)” is a *patient* of the *action* “开放 (open)”).

1.2 Knowledge Acquisition

To complete the abovementioned processes, one can build an automatic system based on manually designed rules inspired by linguistic knowledge. Even though this type of system does not require expensive annotated data, it is still prohibitive to design all the rules manually. In addition, the system designed for a certain language is unsuitable for another unless the rules are re-designed. For a more robust multilingual system, machine learning approaches are always preferred over rule-based approaches in NLP. A certain scale of training data is needed to train a model which contains a large number of parameters indicating different kinds of statistical information such as the tendency of the word "ball" grammatically depending on the word "hit". In real world situations, a system may fail to indicate the dependency relation between a verb "hit" and its object "buzzer" in a more complex structure due to lack of such lexical information in training data. Ambiguity is regarded as one of the biggest problems in NLP. Both syntactic and semantic ambiguities make text harder to analyze. In the sentence "John hit the ball with a bat", a system can hardly tell that the prepositional phrase "with a bat" is modifying the verb "hit" or the object "ball", if there are few such patterns within the training data. In order to compensate for insufficient data, large-scale knowledge extracted from other sources (text corpus), which contains a massive amount of useful information, is considered to be indispensable for automatic text analysis. The term large-scale indicates that the knowledge extracted is quite extensive, coverage-wise. Intuitively, building a real-life question answering (QA) system based on NLP and logical reasoning also requires large stocks of world knowledge.

Manual construction of such knowledge has been widely studied. A number of projects have manually formalized large stocks of knowledge for specific purposes. However, manually constructing and updating such knowledge bases is very time consuming. Besides, manual construction also suffers from low coverage and small scale. As a result, manual knowledge can often be applied to solve the problems within a certain domain. Also it is impractical to use a manual approach to construct knowledge in multiple languages. These drawbacks lead to bottlenecks in the process of knowledge acquisition.

Therefore, recent studies have been focusing on automatic knowledge acquisition. Researchers started looking for methods of acquiring necessary knowledge from large

volume of unprocessed natural language text. These methods usually utilize fundamental NLP techniques such as morphological analysis and syntactic parsing. The data explosion phenomenon in the information age makes it more promising to use large-scale raw texts crawled from the Web. Web texts are relatively easy to obtain and contain knowledge from variety of domains. Web texts undergo constant modification just like languages do. Information such as new words in different languages, new usages of different words can be also captured from the Web texts. We propose an automatic and language-independent framework for knowledge acquisition. This approach can be easily applied to different languages despite the divergence between different languages with respect to morphology and grammar.

1.3 Main Problems

Our main objective is to use knowledge acquired from Web text to improve the quality of dependency parsing and semantic role labeling. We mainly focus on the knowledge that can clarify the relations between predicates and their arguments. The specification of the knowledge basically follows the style of Japanese case frames [21], in which arguments are collected for each predicate from the corpus. Each argument is assigned its corresponding case slot (e.g., "が" case, "を" case, "に" case). These cases indicate different grammatical usages and syntactic roles of the arguments. When we apply this specification for the languages (English, Chinese) without explicit case markers, it becomes problematic to represent each argument of a given predicate. Instead of using the non-terminal tags in a constituent tree which are less representative, we utilize the syntactic roles in dependency trees for those languages without explicit case markers.

For the construction of such knowledge, the performance of the preceding processes, such as the accuracy of dependency parsing etc., is vital. To obtain an accurate dependency parser, a large treebank is essential. However, such training data usually involves a great amount of human labor (annotation of the data) and thus have a size limitation. Besides, training data such as treebanks are always domain specific (e.g., Penn Chinese Treebank and Penn Treebank are mostly composed by the text from newspapers). Therefore, it is also difficult for a parser parse the texts from different domains.

Although, by using such training data, the accuracy of state-of-the-art syntactic parsers

(always evaluated by the evaluation data from the same domain) for languages like English or Japanese is over 90%, it is still not high enough to acquire precise knowledge. Furthermore, if one tries to apply a method of knowledge acquisition to some difficult-to-analyze languages like Chinese and Arabic, the quality of the resulting knowledge will be much worse. When we apply this kind of parsing models to real-world text analysis (e.g., Web text analysis), a bigger performance drop is inevitable. This is caused by not only the domain diversity problem of training data, but also the fact that people always use different expressions or even different grammatical constructions in informal communication (e.g., forum, personal blog etc.)

If all such dependency parses are used for knowledge acquisition, they produce a noisy knowledge base, which leads to the deterioration of subsequent tasks using the knowledge base. How to filter out noise for better quality knowledge acquisition is another issue that is addressed in this thesis. Since dependency parsing plays a fundamental role in knowledge acquisition, we propose to automatically select those dependencies with high quality for knowledge compilation.

1.4 Framework Overview

In order to make use of large-scale raw text that is relatively easy to acquire, we propose a framework that can extract and construct high-quality knowledge from a raw corpus. During the dependency parsing process, for example, a dependency parser tends to judge certain types of dependency relations with high accuracy. On the other hand, some specific types of dependency structures are relatively difficult for a parser to analyze correctly. As a result, a parser will produce automatic parses in different quality according to different properties of dependencies. Instead of using all the automatic parses, it is possible to use only high-quality dependencies for knowledge acquisition. In this thesis, we present a framework for knowledge construction from high quality dependencies that are selected from automatic dependency parses. Figure 1.2 shows the overview of proposed framework. For the raw text, we apply dependency parsing and acquire automatic dependency parses, which inevitably contain parsing errors. Therefore, a high-quality dependency selection approach is proposed to filter those erroneous automatic parses and only select those with high reliability.

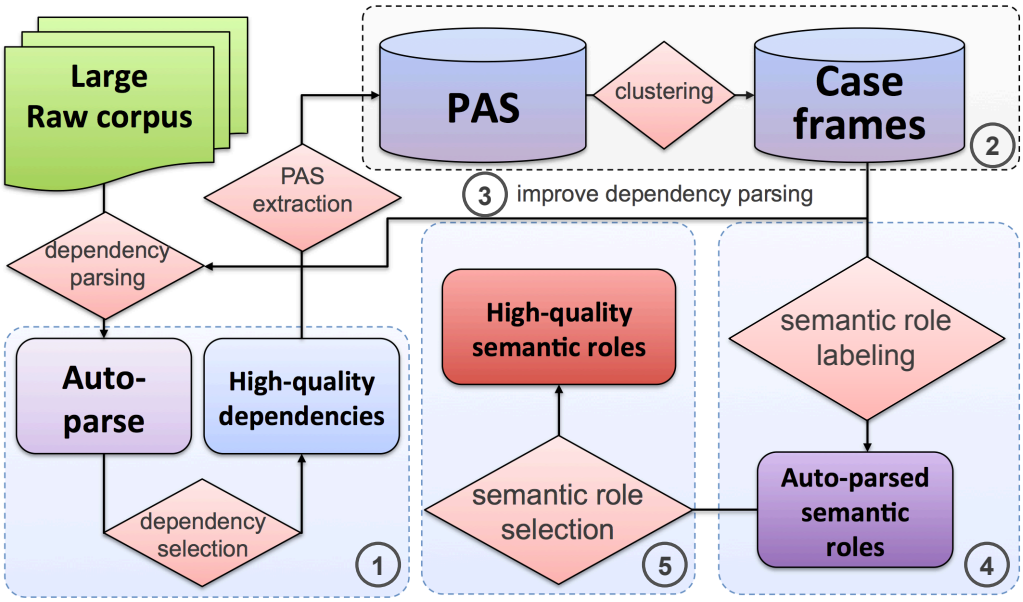

Figure 1.2: Framework overview

From the high-quality dependency parses, we extract predicate-argument structures (PAS) which contain the general idea of a sentence. PAS can be used as an additional knowledge for other NLP tasks. Afterwards, semantic clustering is applied to merge all the predicate-argument structures with similar meaning. The clustered predicate-argument structures are called case frames, which is regarded as another type of knowledge. It is a type of semantic frame that can distinguish between the different usages of each predicate.

PAS knowledge is then used to improve dependency parsing itself. Also we use case frames to improve semantic level analysis which, in this thesis, is referred to as Semantic Role Labeling (SRL).

Errors which are naturally present in automatic analyses are always propagated from the lowest to the highest level. Therefore, auto-labeled semantic roles are also considered to be erroneous. We apply a similar selection approach to select high-quality semantic roles for deep case predicate-argument structures extraction and deep case frame construction.

1.5 Outline of the Thesis

The rest of this thesis is organized as follows: In Chapter 2, we describe an approach that deals with erroneous automatic analysis. This approach automatically selects high-quality syntactic parses in order to suppress the noise problem in knowledge construction.

In Chapter 3, we introduce a framework for high-quality case frames construction. Case frames are a type of knowledge composed of syntactic parses.

In Chapter 4, we use case frames to support dependency parsing.

In Chapter 5, we apply case frames to SRL which is considered to be a type of semantic level analysis.

In Chapter 6, we propose an approach to automatically select high-quality semantic roles from automatic analysis in order to construct more representative knowledge.

Chapter 7 concludes this thesis with some abstract level observations and comments.

Chapter 2

Language-independent Approach to Hiqh-quality Dependency Selection from Automatic Parses

Many knowledge acquisition tasks are tightly dependent on fundamental analysis technologies, such as part of speech (POS) tagging and parsing. Dependency parsing, in particular, has been widely employed for the acquisition of knowledge related to predicate-argument structures. For such tasks, the dependency parsing performance can determine quality of acquired knowledge, regardless of target languages. Therefore, reducing dependency parsing errors and selecting high quality dependencies is of primary importance. In this study, we present a language-independent approach for automatically selecting high quality dependencies from automatic parses. By considering several aspects that affect the accuracy of dependency parsing, we created a set of features for supervised classification of reliable dependencies. Experimental results on seven languages show that our approach can effectively select high quality dependencies from dependency parses.

2.1 Introduction

Knowledge acquisition from a large corpus has been actively studied in recent years. Fundamental analysis techniques have been applied to a corpus and knowledge is acquired

from the analysis. In particular, dependency parsing has been used for tasks like case frame compilation [21], relation extraction [43], and paraphrase acquisition [16]. For these tasks, the accuracy of dependency parsing is vital. Although the accuracy of state-of-the-art dependency parsers for some languages like English and Japanese is over 90%, it is still not high enough to acquire accurate knowledge. Furthermore, if one tries to apply this method of knowledge acquisition to difficult-to-analyze languages like Chinese and Arabic, the quality of the resulting knowledge worsens.

During the dependency parsing process, even if a parser's average performance is high, certain types of dependency relations are judged with high accuracy but other types with very low accuracy. In addition, some types of dependency structures are relatively difficult for any parser to correctly analyze. As a result, a parser will tend to produce automatic parses of varying levels of quality, depending on the properties of dependency. Using full structures of automatic parses for subsequent tasks, such as the extraction of predicate-argument structures, will inevitably lead to noisy results. In practice, however, several tasks do not require full parse structures but only partial ones [12]. To avoid the propagation of errors from automatic analyses, it is preferable to use only high quality dependencies for knowledge acquisition rather than automatic parses. In this study, we present a supervised language-independent approach for selecting high quality dependencies from automatic parses. This method considers linguistic features that are related to the level of difficulty inherent in dependency parsing. We use a single set of dependency labeled data, such as Treebank, part of which is used to train a dependency parser. We conduct experiments on seven languages, five of which are Indo–European languages and two non-Indo–European languages (Chinese and Japanese). The experimental results show that for all the languages, our proposed method can select high quality dependencies than baseline methods.

This chapter is organized as follows. Section 2.2 reviews some research relevant to our approach. Section 2.1 describes the high quality dependency selection process. Section 2.4 presents a detailed description of our research, conducted using three languages, along with the results. Section 2.5 describes several additional experiments conducted using other languages. Section 2.6 presents a discussion of our results. Finally, Section 2.7 presents a conclusion of our approach and proposes future work.

2.2 Related Work

There have been several approaches devoted to automatic selection of high quality parses or dependencies. According to selection algorithms, they can be categorized as supervised and unsupervised methods.

Supervised methods primarily focus on the construction of a machine learning classifier and predict the reliability of parses or dependencies on the basis of various syntactic and semantic features. Yates, Schoenmackers, and Etzioni (2006) created WOODWARD, which is a Web-based semantic filtering system. They first mapped the parses to a logic-based representation called *relational conjunction* (RC). Thereafter, four different methods were employed to analyze whether a conjunct in the RC representation was likely to be reasonable. Kawahara and Uchimoto (2008) built a binary classifier that determines the reliability of each parse. The linguistic features they used for the classification, such as sentence length, the number of unknown words, and the number of commas, are based on the idea that the reliability of parses is determined by the degree of sentence difficulty. The work most related to ours is that of Yu, Kawahara, and Kurohashi (2008). They proposed a framework that selects high quality parses in the first stage and then high quality dependencies from those parses. In comparison with their work, we consider that even some low quality sentences can contain high quality dependencies. In addition, we take into account other characteristics that can directly affect high quality dependency selection, such as context information and tree features.

Among supervised methods, ensemble approaches were also proposed. For example, Reichart and Rappoport (2007) judged parse quality using a Sample Ensemble Parse Assessment (SEPA) algorithm. They trained several different parsers by using samples from the training data. Thereafter, the level of agreement among those parsers was used to predict the quality of the parse. Another similar approach proposed by Sagae and Tsujii (2007) also selected high quality parses by computing the level of agreement among different parser outputs. However, different from the research previously mentioned, which used several constituency parsers trained with different training data sets, they used a single data set to train different dependency parsing algorithms. Different from the abovementioned methods, our method judges the reliability of each dependency produced by a parser.

Unsupervised algorithms for detecting reliable dependency parses have been proposed. Observing the score of each edge is a basic method for judging the reliability of each dependency [17]. Goldberg and Elhadad (2012) proposed a method of assigning each edge a risk score as the inverse of the confidence score defined by a risk function. However, selecting dependencies purely on the basis of such scores is not sufficient for judging high quality dependencies. We reached this same conclusion in our comparative evaluations of dependency selection approaches on different languages. Reichart and Rappoport (2009) proposed an unsupervised method for high quality parse selection, based on the idea that syntactic structures frequently produced by a parser are more likely to be accurate than those produced less frequently. They developed POS-based Unsupervised Parse Assessment Algorithm (PUPA) to calculate the statistics summarizing the POS tag sequences of parses produced by an unsupervised constituency parser.

Dell'Orletta, Venturi, and Montemagni (2011) proposed ULISSE (Unsupervised LInguiStically driven Selection of dEpendency parses), which is also an unsupervised system. Different from PUPA, this method addressed the reliable parse selection task using an unsupervised method in a supervised parsing scenario. Rather than using constituency-related features, such as ordered POS tag sequences, they used dependency-motivated features, such as parse tree depth and dependency link length. Although unsupervised methods may solve the domain adaption issue and do not use costly annotated data, the accuracy of selected parses, which is less than 95%, still needs to be improved for knowledge acquisition tasks.

2.3 High-quality Dependency Selection

In this section, we present a framework for highly reliable dependency selection from automatic parses. Figure 2.1 shows the overview of our approach. We use a part of a treebank dataset to train a parser and another part to train a binary classifier that judges the reliability of a dependency. We use support vector machines (SVMs) for this classification.

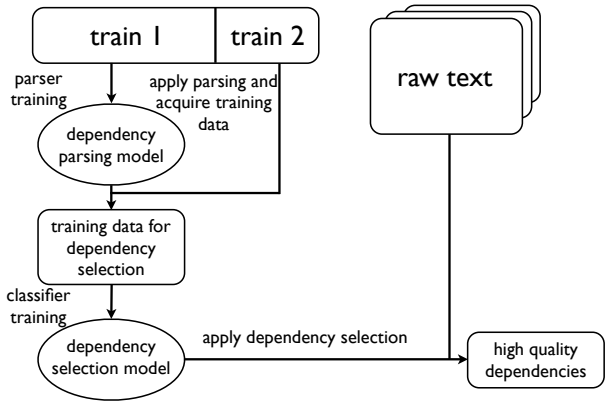

Figure 2.1: Overview of high-quality dependency selection.

2.3.1 Training Data for Dependency Selection

Supervised methods always require manually annotated training data that are usually very expensive to obtain. Owing to limited existing resources, we train a classifier for selecting highly reliable dependencies from parsing outputs using training data from the same treebank that is used in the first stage dependency parsing. In particular, the standard training section of the treebank is used to train a dependency parser and then the development section is used to apply dependency parsing using the previously trained parser. From the output parses of the development section, we acquired training data for dependency selection by collecting each dependency. We then judge the success of each dependency according to the gold standard data. All correct dependencies were used as positive training examples for dependency selection and vice versa.

2.3.2 Dependency Selection

We judge the dependency in each parse and retain only high quality output for knowledge acquisition. There are many factors that affect the parsing performance, such as distance between dependencies and complexity of tree structures. By taking these factors into consideration, we create sets of features for selecting high quality dependencies. Tables 2.1, 2.2, and 2.3 present the details of these features.

Feature	Description
POS_{head},POS_{mod}	Part of speech pair of head and modifier
$Word_{head}$,$Word_{mod}$	Word pair of head and modifier
Distance	Distance between the head and its modifier
HasComma	If there exists a comma between head and modifier, set as 1; otherwise set as 0
HasColon	If there exists a colon between head and modifier, set as 1; otherwise set as 0
HasSemiColon	If there exists a semi-colon between head and modifier, set as 1; otherwise set as 0

Table 2.1: Basic features for dependency selection.

Basic Features

If there is a comma, colon or semi-colon between two words, they are much less likely to have a dependency relation than those pairs that do not contain any punctuation. It is much more difficult for parsers to correctly analyze dependencies that contain punctuation than those without punctuation. We use the most common punctuations as features for dependency selection. A dependency relation between words is much more likely when arguments are nearby [34]. Therefore, distance between words is also an important factor that reflects the difficulty of judging dependency relations. Yu et al. (2008) used the abovementioned features but did not use $Word_{head}$, $Word_{mod}$, or the context features, which are described in the next section.

Context Features

In addition to these basic features, we consider context features that are thought to affect parsing performance. Table 2.2 lists these context features. For example, the two sentences "they eat salad with a fork" and "they eat salad with sauce" contain the PP-attachment ambiguity problem, which is one of the most difficult problems encountered in parsing. The two prepositional phrases "with a fork" and "with sauce" depend on the verb "eat" and the noun "sauce," respectively. However, a dependency parser can hardly resolve these two cases. Therefore, we tend to judge this type of structure as unreliable. Consider another similar sentence "they eat it with a fork." Because the prepositional phrase "with a fork" cannot depend on the pronoun "it" but only on the verb "eat," this case can be clearly judged as a highly reliable dependency. In some more complex cases, it is also necessary to observe a larger span of context. To learn such linguistic characteristics automatically, besides POS tags and the head and modifier in a dependency, we also use the preceding and following one and two words, respectively, along with the POS tags of the abovementioned linguistic characteristics.

Another important fact is that verb phrases in the dependency tree structure of a parse are normally the root node of the entire dependency tree or the parent node of a subtree. When a word pair contains a verb phrase between them, the two words are always on different sides of the parent node. Thus, these kinds of word pairs normally have no dependency link between them. For example, in SVO (subject-verb-object) languages such as English and Chinese, the subject appears first, the verb second, and then the object third. The most common example of this is when the subjects and objects located on both sides of the verb are the modifiers of the verb. Therefore, argument pairs that have a verb between them rarely have a dependency relation. Observing whether there are verb phrases between head-modifier pairs can help judge dependency reliability.

Tree Features

The input for our high quality dependency selection method is a dependency tree. It is natural to use tree features to identify dependency quality. On the basis of a head-modifier dependency, we observe modifiers of a modifier, i.e., children nodes, a head's parent node–which are called grandparent nodes–and children nodes of the grandparent

Feature	Description
HasVerb	If there exists a verb between head and modifier, set as 1; otherwise set as 0
$POS_{head-1/mod-1}$	Part of speech tag of the preceding word of head and modifier
$POS_{head+1/mod+1}$	Part of speech tag of the following word of head and modifier
$Word_{head-1/mod-1}$	The preceding word of head and modifier
$Word_{head+1/mod+1}$	The following word of head and modifier
$POS_{head-2/mod-2}$	Part of speech tag of the preceding second word of head and modifier
$POS_{head+2/mod+2}$	Part of speech tag of the following second word of head and modifier
$Word_{head-2/mod-2}$	The preceding second word of head and modifier
$Word_{head+2/mod+2}$	The following second word of head and modifier

Table 2.2: Context features for dependency selection.

Feature	Description
POS_{Grand}	Part of speech tag of grandparent node of a modifier
$Word_{Grand}$	Word of grandparent node of a modifier
$POS_{Luncle/Runcle}$	Part of speech tag of the leftmost/rightmost uncle node of a modifier
$Word_{Luncle/Runcle}$	Word of the leftmost/rightmost uncle node of a modifier
$POS_{Lchild/Rchild}$	Part of speech tag of the leftmost/rightmost child node of a modifier
$Word_{Lchild/Rchild}$	Word of the leftmost/rightmost child node of a modifier

Table 2.3: Tree features for dependency selection.

node, which we call uncle nodes.

Edge Score

Some dependency parsers output the score of each dependency (i.e., edge confidence value) during the parsing process. A high score indicates a high possibility that the dependency is correct. However, utilizing this score as the only feature is not sufficient for acquiring high quality dependencies, especially in low quality parses. We consider the real value of the score as an additional feature.

2.4 Main Experiments

2.4.1 Experimental Settings

We first conducted our experiment on three languages, including English, Japanese, and Chinese, using the data from the CoNLL-2009 shared task [15]. For each language, we employed the MSTparser[1] (version 0.5.0) as a base dependency parser and use the training data to train a dependency parsing model. KD-Fix (0.05 * 20) confidence score [37] is one of the edge score calculation methods in the MSTparser, which was reported as the best method for the MSTparser to predict edge scores. We used the KD-Fix value as the edge score in all experiments. We applied the development data to the dependency parsing model to acquire the training data for dependency selection. We used automatic POS tags for the dependency selection approach (automatic segmentation for Japanese and Chinese). The MXPOST[2] tagger was used for English automatic POS tagging, and for Chinese, we employed MMA [29] [4] to apply both segmentation and POS tagging. We used JUMAN[3] for Japanese morphological analysis. Then KNP[4] is utilized for Japanese dependency parsing.

From the dependency parser output, we collected training data for high quality dependency selection. All correct dependencies, according to the gold standard data, were defined as positive examples and vice versa. We utilized SVMs to complete this binary classification task, specifically, we employed SVM-Light[4] with a linear kernel. The option -j *ratio* was used to solve the positive and negative imbalance in the training data for the classifier, where *ratio* was calculated by dividing the number of negative samples by the number of positive samples.

In order to compare these results with the work done by Yu et al. (2008), we set the basic feature as a baseline. The evaluation data for each language was then used to evaluate the effectiveness of dependency selection.

[1] http://www.seas.upenn.edu/~strctlrn/MSTParser/MSTParser.html
[2] http://www.inf.ed.ac.uk/resources/nlp/local_doc/MXPOST.html
[4] http://chasen.org/~taku/software/darts/
[3] http://nlp.ist.i.kyoto-u.ac.jp/EN/index.php?JUMAN
[4] http://nlp.ist.i.kyoto-u.ac.jp/EN/index.php?KNP
[4] http://svmlight.joachims.org/

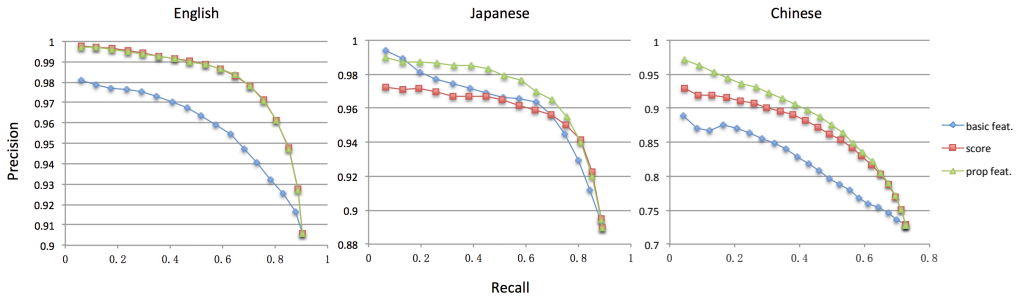

Figure 2.2: Precision-recall curves of selected dependencies for English (left), Japanese (middle) and Chinese (right).

2.4.2 Evaluation Metrics

On the basis of the output of the SVMs, we selected dependencies with output scores greater than a specific threshold, with a higher output score indicating a more reliable dependency. As a result, a high threshold meant a low recall. Thereafter, we evaluated the selected dependencies by calculating the percentage of correct head-modifier dependencies (excluding punctuations) according to the gold standard data. Precision and recall were calculated as follows.

$$precision = \frac{\#\,of\ correctly\ retrieved\ dependencies}{\#\,of\ dependencies\ retrieved}$$

$$recall = \frac{\#\,of\ correctly\ retrieved\ dependencies}{\#\,of\ correct\ dependencies\ in\ gold\ standard\ data}$$

In automatically tagged and parsed Chinese and Japanese data, there were segmentations that were incorrectly produced. These cases were treated as incorrect instances. Note that the maximum recall for each language was equal to the precision of the base parser.

2.4.3 Experimental Results

Effectiveness of Dependency Selection

Figure 2.2 shows the precision-recall curves of the selected dependencies using SVMs. Different criteria were used for all languages. The curves labeled “basic feat.” indicates

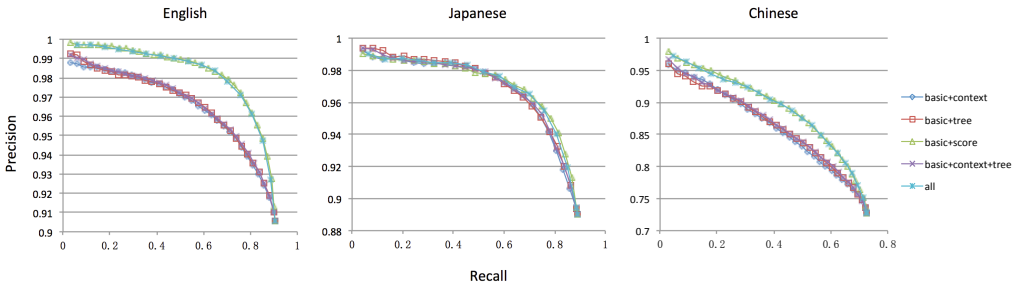

Figure 2.3: Precision-recall curves of selected dependencies using different features for English (left), Japanese (middle) and Chinese (right).

that basic features were used for the dependency selection. "Score" is the method that selects dependencies with the edge scores (MSTparser's KD-Fix) higher than the threshold. Note that this method does not use SVMs. "Prop feat." indicates that proposed features were used, in which the real values of the edge scores were used as a feature in the classifier. We can see from the results that our proposed features outperformed the other feature sets in most cases, especially for Chinese and Japanese.

To investigate the feature that is most effective on the dependency selection, we plotted different precision-recall curves using different feature combinations (Figure 2.3). Note that "all" is equal to our proposed method in Figure 2.2, which uses all features. We can see that all features work differently for different languages. For example, edge scores can effectively help select high quality dependency for English but context and tree-based features perform better on Japanese. Those languages have different base parser's performance and different dependency styles (e.g., head-final for Japanese). We speculate that features such as distance between arguments and comma have a significant influence on Japanese dependency selection.

Statistics of Selected Dependencies

In this section, we investigate the distribution of dependencies in order to determine the primary types selected. This investigation lets us observe whether selected dependencies are biased towards certain types of POS (e.g., meaningless patterns, such as "DT NN"). Each dependency type was represented by coarse-grained POS pairs (the first two characters of the POS names). Figures 2.4 and 2.5 summarize the statistics for selected

dependency POS pairs using different methods for different languages. In each figure, the left and middle graphs represent the dependencies selected under different thresholds (i.e., 50% and 20% recall, respectively), and the rightmost graphs are plotted without the dependency selection.

For all languages, under both selection methods (i.e., proposed method and the selection method by edge score), dependencies with nouns were dominant across all types. For large-scale knowledge acquisition, however, dependencies with verbs are most important because verb phrases always contain most of the information about a specific event. Therefore, verb phrase extraction is key in recognizing predicate-argument structures in knowledge acquisition. The pattern "NN VB" also was prominent, which indicates that predicate-argument dependencies were still selected quite frequently among the high quality dependencies.

Selecting dependencies using different features will inevitably lead to the loss of some informative patterns along with those that are considered noisy. From the results, both selection methods have similar tendencies in selecting various types of dependencies. The key point here is that our proposed method, which used different types of features, still produced a high proportion of informative dependencies.

2.5 Additional Experiments

2.5.1 Experiments on Other Languages

We applied additional multilingual experiments on Catalan, Czech, Spanish, and German using the CoNLL-2009 shared task data. The MSTparser again was used for dependency parsing. Note that these languages possess the non-projective property, which indicates crossing edges in a dependency tree. They allow for more constructions than the projective constraint. As a result, the non-projective option in MSTparser was triggered. Figure 2.6 shows the precision-recall curves of dependency selection for these four languages. We directly used the 6th column in the CoNLL-2009 shared task data as automatic POS tags.

For each language, we exhibited the precision at various recall values in order to highlight the quality of selected dependencies under different selection thresholds. Table 2.4 and Table 2.5 show the precision of dependencies selected by our method under

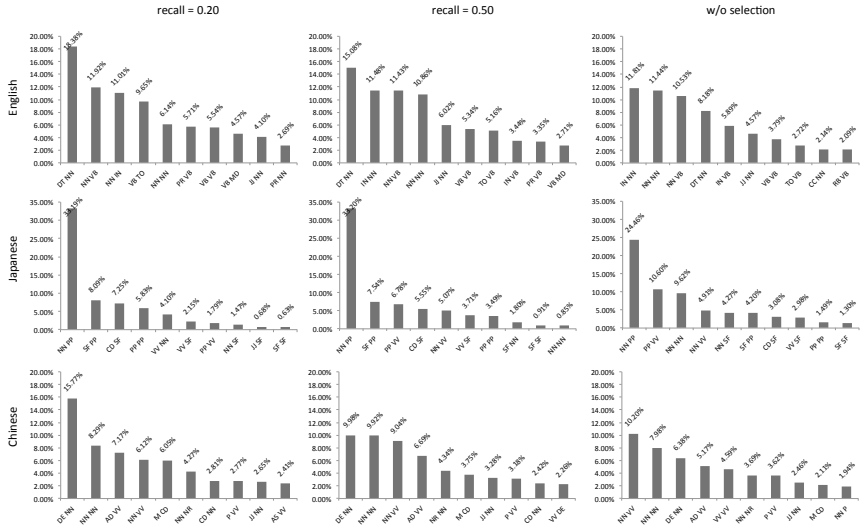

Figure 2.4: Statistics summarizing dependency POS tags that use the proposed method: dependencies without selection (right), dependencies with 50% recall (middle), dependencies with 20% recall (left).

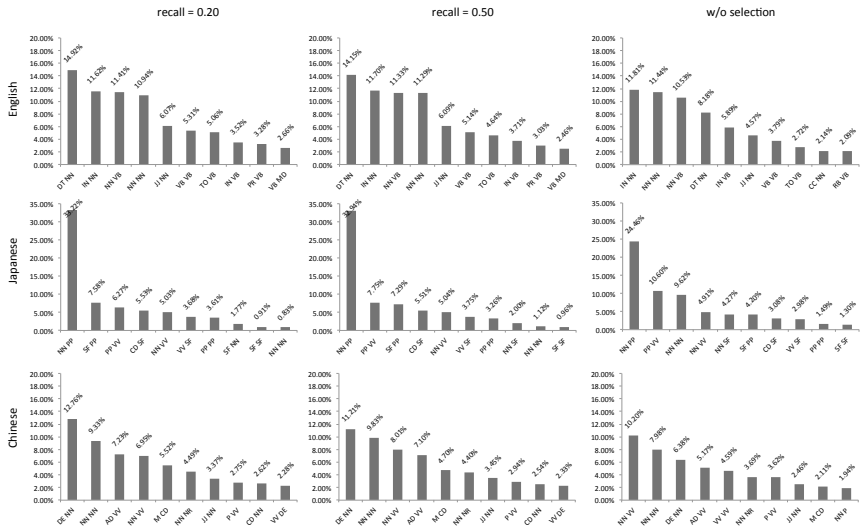

Figure 2.5: Statistics summarizing dependency POS tags that use edge scores: dependencies without selection (right), dependencies with 50% recall (middle), dependencies with 20% recall (left).

		English	Japanese	Chinese
base parser		0.906	0.877	0.825
recall=20%	basic feat.	0.976	0.980	0.870
	score	0.996	0.972	0.911
	prop feat.	0.996	0.988	0.944
recall=50%	basic feat.	0.965	0.967	0.796
	score	0.990	0.966	0.854
	prop feat.	0.990	0.981	0.864

Table 2.4: Precision of selected dependencies under different criteria (JCE)

20% and 50% recall for all languages. From the results, we can see that the dependency selection method that used our proposed features outperformed the method that used the basic features for all languages. For Japanese, Chinese, and Catalan, using the proposed features had a greater effect on high quality dependency selection than other languages, compared to the method only using the edge scores.

2.5.2 Experiment on Different Domain

One of the biggest problems that most data-driven parsers face is the domain adaptation problem in that their accuracy decreases significantly owing to the lack of domain-specific knowledge. We applied the dependency parsing model trained on section 2 to section 21 of the Penn Treebank (PTB) to the Brown corpus, and obtained an unlabeled attachment score of 0.832, which is significantly lower than the in-domain score by 7.4%.

We applied the same dependency selection model trained on the PTB training sections to the pBrown corpus. Figure 2.7 shows the precision-recall curves for dependency selection on the PTB test section (section 23) and the Brown corpus using different selecting methods. Similarly to previous figures, “brown basic” indicates selection using basic features. “Brown score” indicates only edge scores were used for classification. “Brown prop.” and “ptb prop.” represent proposed features were used for dependency selection on brown corpus and PTB, respectively. From the results, we can see that our proposed method outperforms the one that used basic features, and also has an advantage

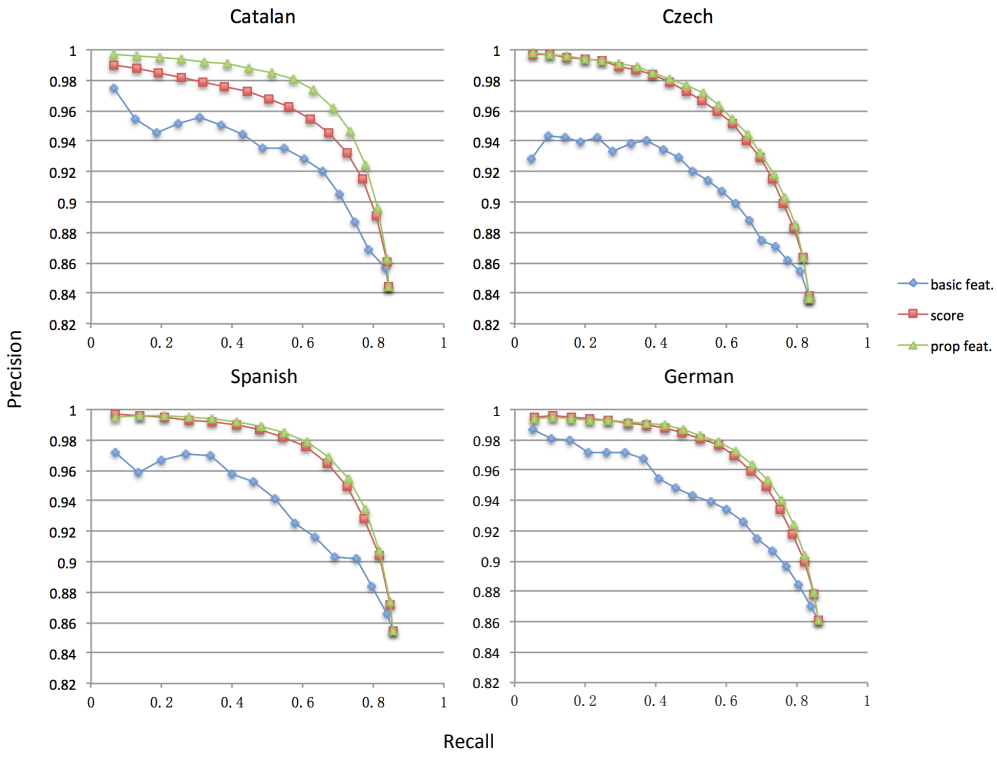

Figure 2.6: Precision-recall curves of dependency selection in four different Indo-European languages: Catalan (top left), Czech (top right), Spanish (bottom left), and German (bottom right).

		Catalan	Czech	Spanish	German
base parser		0.845	0.836	0.855	0.861
recall=20%	basic feat.	0.945	0.938	0.966	0.972
	score	0.985	0.994	0.995	0.994
	prop feat.	0.995	0.994	0.996	0.994
recall=50%	basic feat.	0.935	0.921	0.945	0.943
	score	0.968	0.971	0.985	0.983
	prop feat.	0.985	0.976	0.987	0.985

Table 2.5: Precision of selected dependencies under different criteria (Euro.)

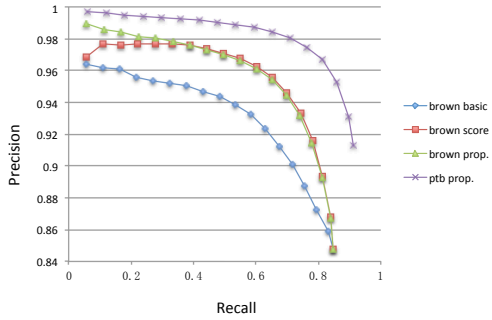

Figure 2.7: Precision-recall curves of dependency selection from the Brown corpus.

over the method that used edge scores with low recall. For examples, when the recall was 20%, high quality dependencies with a precision greater than 98% could be acquired. This shows that our method works well on data from different domains and that there is a way to acquire knowledge from a large raw corpus in different domains (e.g., the Web).

2.5.3 Experiment using Different Proportions of Training Data

To determine the amount of training data required to achieve a reasonable dependency selection performance, we used different proportions of the PTB development section to train different classifiers. Thereafter, we used the same test set to evaluate each classifier using the same metric described in Section 2.4. Note that the method that uses only the

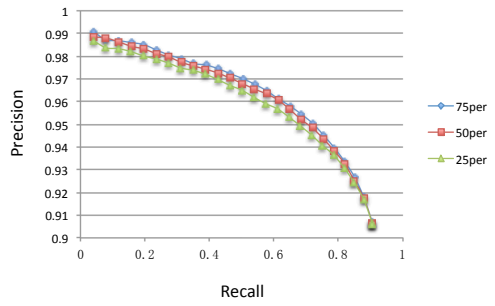

Figure 2.8: Precision-recall curves of dependency selection obtained using training data sets of various sizes.

edge scores does not change with different data proportions (i.e., one should get similar classification performance using the edge score as a feature regardless of the size of the training data set). Thus, we only used features including "basic", "context", and "tree" in this experiment. Figure 2.8 shows the precision-recall curves of dependency selection obtained using different sized training data sets. The performance decreases slightly when the training data set decreases in size. However, a precision greater than 98% is still achievable when the recall is 20%, even when using only 25% of the training data. From the results, we can see that training data sets of various sizes can be used to train effective classifiers.

2.5.4 Experiment using a Different Parser

As MSTparser is a graph-based dependency parser, we wanted to further test our proposed framework on different types of syntactic parsers. Therefore, we choose the Stanford parser[5], a probabilistic context-free grammar (PCFG) parser. Although the output of the Stanford parser can be converted into dependency style, it is unable to output an edge score. As a result, we used features including "basic", "context" and "tree" in this experiment. We conducted the experiment on English and Chinese. We directly used English raw text and Chinese segmented text by MMA as input, and employed the Stan-

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

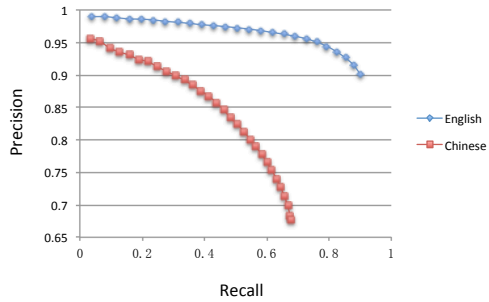

Figure 2.9: Precision-recall curves of dependency selection obtained using the Stanford parser.

ford parser for POS tagging and parsing. Figure 2.9 shows the precision-recall curves for English and Chinese. English can achieve high precision around 98.5% when the recall is 25%. Even though the base parser for Chinese achieved only 67.5%, we can still extract dependencies with over 92% precision.

2.5.5 Using Dependency Selection in Other Tasks

We applied our proposed dependency selection approach to predicate-argument structure extraction and distributional similarity calculation. First, we applied the dependency selection to automatic parses and used only high quality edges to extract predicate-argument structures. With a central focus on verbs in each dependency tree, we used only verb-dependent arguments and represented each argument by its syntactic surface case (i.e., subject, object, prepositional phrase, etc.) with sets of heuristic conversion rules. Distributional similarity is a method that determines word similarity on the basis of a metric derived from the distribution of the verb and its arguments in a large text corpus [30].

For English, we used a large-scale Web corpus that contained 200 million sentences. We employed the Wordsim353[6] data set for evaluation, which contains human-assigned similarities between each word pair. For Chinese, five million sentences from the Chinese

[6] `http://alfonseca.org/eng/research/wordsim353.html`

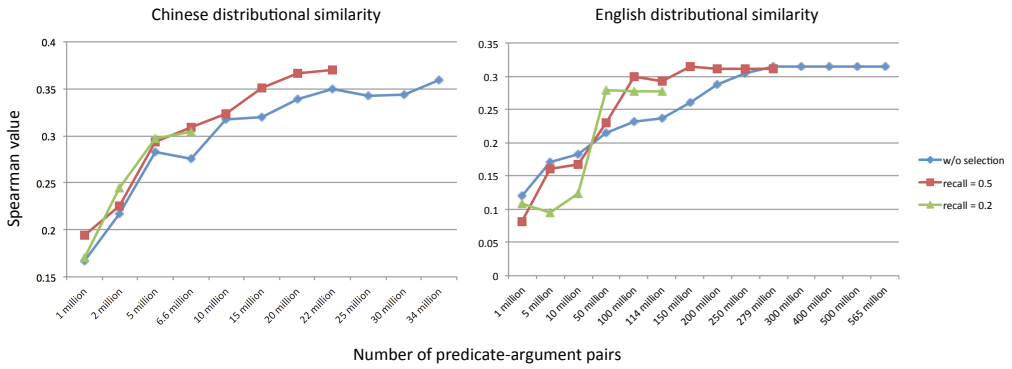

Figure 2.10: Spearman values under different selection thresholds

Gigaword were used for distributional similarity calculation. We also used a manually constructed gold-standard data set[7] containing more than 500 word pairs for Chinese word similarity evaluation.

For each word pair in the evaluation set, we evaluated the word similarity using the distributional thesauri calculated using the acquired predicate-argument pairs. We used spearman's correlation coefficient, calculated by comparing the ranks between the two sets of similarities (i.e., gold similarities, and automatically calculated similarities), to evaluate the quality of the thesauri.

Data size is an important factor that can affect distributional similarity calculations. Therefore, to compare the thesauri calculated from predicate-argument pairs of various sizes, we randomly sampled different sized sets of predicate-argument pairs. Figure 2.10 shows the Spearman coefficient values for the distributional similarity calculations under the three criteria mentioned above. As we can see, performance of distributional similarity calculations can be improved by selecting high quality dependencies, for both Chinese and English. Using the dependency selection process, the negative effects that the noisy data can have on semantic representation of word similarity can be effectively reduced. The result shows that the quality of dependencies and data size are both important factors for distributional similarity calculation.

[7] http://www.cs.york.ac.uk/semeval-2012/task4/

2.6 Discussion

We achieved dependency precisions of 99.6%, 97.8%, and 98.8% for English, Chinese, and Japanese, respectively, using automatically tagged data with a recall of 20%. All four European languages have around 99.5% precision under this recall. Our proposed features show a significant advantage over the original feature set proposed in the previous study of Yu et al. (2008). By taking into account context and tree information, we can effectively help the system learn automatic dependency parse reliability not only from the same domain but also from other domains. As shown in the experiments, the results are quite promising for different types of syntactic parsers, as well.

The statistics calculated for selected dependencies showed that even when we adopted a low recall value to obtain a high precision they contained many dependencies related to nouns and verbs. The low recall (e.g., 20%) can be compensated for by using very large raw corpora, which are relatively easy to acquire. The applicability to different domains also allows us to acquire knowledge from large raw corpora of various domains. Moreover, our proposed approach can benefit subsequent NLP tasks, such as distributional similarity calculation.

Each feature type had a different influence on different languages. For example, English and Chinese performed similarly when using edge scores vs. proposed features. However, Japanese showed better performance using our proposed features. We speculate that features such as the distance between arguments and commas had a significant influence on Japanese dependency selection. Even though selection using edge scores can help acquire high quality dependencies, which sometimes have similar performance to our proposed features, it is very important to note that edge scores are not always available for different types of parsers (e.g., Stanford parser). Furthermore, producing edge scores (e.g., KD-Fix of MSTparser) demands a larger amount of computer memory, which normally makes it impractical to apply parallel analyses to large-scale corpora in practice.

2.7 Summary

In this chapter, we proposed a classification approach for high quality dependency selection. We created a set of features that consider context and tree information in selecting

highly reliable dependencies from parsed sentences. This approach can extract high quality dependencies even from low quality parses. The experiments showed that our method works for in-domain parses as well as out-of-domain parses.

We can extract high quality dependencies from a large corpus such as the Web, and can subsequently assist knowledge acquisition tasks, such as subcategorization frame acquisition [28] and case frame compilation [22], which depend highly on the parse quality.

Since automatic parses can be used to improve the base parser itself [5], we also plan to use high-quality dependencies to improve dependency parsing.

Chapter 3

A Framework for Compiling High-quality Case Frames From Raw Corpora

In order to realize text understanding by computers, identifying various types of relations in text is a necessary step. Especially, clarifying relations between predicates and their arguments is an essential task.

To precisely capture these relations, a wide-coverage knowledge source which contains this kind of relation is indispensable. One of such knowledge sources is case frames. Case frames can represent relations between a predicate and its arguments and contain basic linguistic knowledge that humans have. Case frames in multiple languages not only can improve fundamental analysis of these languages but also can support multilingual applications, including machine translation.

Manually constructing case frames is very costly not only for construction but also for updating. Furthermore, manually constructed case frames are always suffering from low coverage. Since it is important to compile case frames for multiple languages, this thesis proposes a language-independent framework for case frame construction even though the grammars and characteristics for the languages are quite different.

The main point of this framework is to extract highly reliable predicate-argument structures from large-scale raw corpora to produce case frames. We only create a small

sets of language-specific rules for filtering out unreliable structures in each language.

Both for English and Chinese, we implemented construction experiments. We made use of tagged corpora to evaluate the extracted predicate-argument structures and the precision for both English and Chines achieved over 97%. We also compared the constructed case frames with other knowledge sources and found the our case frame outperform about 30% on average.

3.1 Introduction

In natural language processing (NLP), case frames are a very essential knowledge base which represents the relations between a predicate and its arguments. Case frames can support various types of NLP applications, such as parsing, machine translation, recognizing textual entailment and paraphrasing. For instance, in the classical example of parsing, "saw a girl with a telescope," there is an ambiguity problem of which argument the prepositional phrase, "with a telescope," is modifying. It would be easy to judge that the prepositional phrase belongs to the verb if knowledge of a case frame "see someone/something with telescope," is available.

This kind of dependency ambiguity also occurs in other languages normally. In a case of anaphora, people are always expecting effective ways to indicate the reference expression in the text like "my pet ate an apple because it is hungry". With the assistance of case frames "someone/pet/child is hungry", we could infer that "it" refers to "pet" with much higher confidence rather than the apple.

For NLP applications in multiple languages and multilingual applications, compilation of large-scale case frames in these languages is important. Manually compiling this kind of knowledge in multiple languages would be too costly and would be limited by low coverage.

Even though the grammars in different language are quite dissimilar, we show that extracting predicate-argument structures to build case frames is achieved by one common framework. Although, in the researches of Japanese Case frames, case marker could be a very essential information the assist the construction, for other language such as with no case marker, lack of this information will lead to many difficult problems. Also in different languages, they have their own characteristics which are the reasons of many

parsing errors. For example, the omission of clause marker in English or the ambiguity of character DE in Chinese. We consider to make use the existing resource to improve the performance of the whole framework itself. In this research, we focus on the construction in Chinese and English.

The rest of this chapter is structured as follows. Section 3.2 introduces some related work about case frames construction. Section 3.3 describes the specification of case frames we construct. Section 3.4 introduces the detail of our construction method. Section 3.5 presents our experiments and evaluation and Section 3.6 is the conclusion and sketches our future work.

3.2 Related Work

To assist many kinds of text understanding task and other fundamental analysis, many language resources were built in previous studies. For example, there were two manually constructed language resources called FrameNet and PropBank which are corpora with verbal annotations. People were also working on automatic construction of such knowledge resource such as subcategorization frames. Subcategorization frames indicate the information of predicates and their arguments' category. In later period, Japanese case frames have also been automatically constructed. In this section, we introduce these related studies in detail.

3.2.1 Manually Constructed Frames

FrameNet

FrameNet is a project which is building an annotated corpus of semantic frames in English. FrameNet produces a both human-readable and machine readable database which provides more than 10,000 word senses and more than 170,000 annotated sentences that can be used as a good training dataset in semantic role labeling. As a result, it has become a very important lexical resource for text understanding in NLP.

FrameNet makes use of a theory called frame semantics which is an extension of case grammar. In this theory, people consider that the best way to explain the meaning of words by their semantic frames. A semantic frame is usually a description for the type

of event, relation or entity with its participants. Each semantic frame consists of several frame elements. For example, when describing the concept of "fear", there is always a person or sentient entity that experiences or feels the emotions (experiencer), the body part, gesture, or other expression of the experiencer that reflects his or her emotional state which describes a presentation of the experience or emotion denoted by the adjective or noun (expressor), a person, event, or state of affairs that evokes the emotional response in the experiencer (stimulus) and the general area in which the emotion occurs which can indicate a range of possible stimulus (topic). This kind of elements is called frame elements. Some more complex frames such as "revenge" include more frame elements like "offender", injury, injured_party, avenger and punishment. Other simple ones such as "placing" only contains an agent, a theme which stands for the thing to be placed and a goal which means the location in which it is placed. In the discription of one concept such as fear, some other words like "afraid", "frightened" or "terror" are usually evoked and be named as the lexical unit of the frame of fear. The main purpose of FrameNet is to define the frames and to annotate sentences to show how the frame elements fit syntactically around the word that evokes the frame, as in the following examples of "fear":

$$[_{<Experiencer>}I]\ have\ been\ [_{<Degree>}so]\ frightened$$

$$[_{<Experiencer>}Mame]\ was\ [_{<Degree>}extremely]\ freaked.$$

where the words "frightened" and "freaked" are the lexcial units that evoke the frame "fear". The frame element such as Experiencer describes the person who experiences the emotion. The other frame element like "Degree" is the extent to which the experiencer's emotion deviates from the Norms for the emotion.

Instead of syntax, the frames in FrameNet annotations are basically semantic, which can be often language independent. So no matter how the syntax varies when we describe a concept in different language, we can use some common frame elements. For example, frames about buying and selling always involve the frame elements "buyer", "seller", "good", and "money" etc. Some projects to build FrameNets in other languages have been carried out.

PropBank

Another annotated language resource which is always compared with FrameNet is PropBank, which is called a "proposition bank." PropBank is a corpus annotated with verbal propositions and their arguments. Different from FrameNet, PropBank mainly focuses on verbal information and commits to annotating all the verbs in its data. Additionally, all the arguments to each verb must be syntactic constituents, which means each argument is composed by one word or a group of words that functions as a single unit within a hierarchical structure.

> *[ARG0 Sam]* hit *[ARG1 the ball]*
>
> *[ARG1 The firm]* has been hit *[ARG2 with big financial settlements]*

As the example shown above, PropBank firstly provides consistent argument labels across different syntactic realizations of the same verb. In this example, each argument to the verb "hit" is labeled and numbered. Also, PropBank assgins all the arguments with functional tags such as manner (MNR), locative (LOC) and temporal (TMP).

> *[ARG0 He]* hit *[ARG1 the lanes] [ARGM-TMP three years ago] [ARGM-ADV on the advice of his doctor]*

PropBank annotation also involves finding antecedents for empty arguments which are omitted in the text similar with zero-anaphora task. In the following example, the omission of John is annotated.

> *[ARG0 John-1]* couldn't be bothered *trace*-1 to make it *[ARG1 to his own funeral]*

Even though both of FrameNet and PropBank can provide annotated data as gold standard for many NLP applications, manually constructing this kind of language resource can hardly avoid the fact that the coverage of knowledge in each language repository is relatively low. Both acquiring and updating this kind of database would much more costly than automatic ways.

3.2.2 Automatically Constructed Frames

Subcategorization Frames

In early research, subcategorization frames were proposed to represent the relations between the verbs and other syntactic arguments in the text. Subcategorization frames do not concern the meaning of each argument but focus on the argument patterns or argument categories of the verb, and judge whether a certain kind of pattern makes sense on the frequency of this type pattern extracted from corpora. Take the verb "send" as an example. One of the subcategorization frames of the verb "send" would be "NP send NP PP" and the original sentence could be "I send an E-mail to her". This means the verb 'send' usually takes a noun phrase as a subject, another noun phrase as an object and a prepositional phrase as modifier. In the early stages of NLP, subcategorization frames for English were constructed manually, e.g. the COMLEX Syntax (Grishman et al., 1994) and the ANLT-Data dictionary (Boguraev and Briscoe, 1987). However, manual construction are prone to errors which are difficult to detect automatically.

Automatic acquisition of subcategorization frames have been firstly proposed using simple grammatical regularities to learn lexical syntax (Brent, 1993). Other approaches implemented automatic acquisition of verb subcategorization frames and their frequences from large tagged corpus (Ushioda et al., 1993; Manning, 1993). Another research provides a system which constructed a subcategorization dictionary from textual corpora and proved that a subcategorization dictionary can improve the accuracy of parsing (Briscoe and Carroll, 1997). All of these approaches mainly focus on verbs and do not distinguish ambiguous predicate senses which became one cause of the poor performance. Recently, word sense disambiguation system was employed to guide the acquisition process (Korhonen and Preiss, 2003). Comprehensive subcategorization lexicons are constructed from five corpora and the Web text (Korhonen et al., 2006). However, without giving the detail information of other arguments that surround the verb is the biggest limitation of subcategorization frames and makes it less effective to use subcategorization frames to assist other applications in NLP.

Japanese Case Frames

Although syntactic parsing plays a very important role in NLP, there are many information cannot be efficiently clarified due to the characteristic of different language. In the case of Japanese, because of the special language characteristic such as omission of case components and case marker, case structure analysis is essential and construction of case frame became a very important issue. Different from subcategorization frames, case frames provide not only the information of predicate but also its arguments along with their relations in the text.

Studies of Japanese case frames also began with manual framework (Ikehara et al., 1997). But manually constructed case frames have the same limitation of coverage with other language resources such as FrameNet and Propbank.

Others have proposed ways to automatically acquire Japanese case frames from roughly parsed corpora (Utsuro et al., 1997). But the small size of existing analyzed corpora made it nearly impossible to construct a wide-coverage lexicon, and verb sense ambiguity problem is not resloved. In addition, deep dependence on the thesaurus of semantic hierarchy for nouns also becomes the limatation of their work.

Another research learns case frames for sets of verbs from one year's worth of newspaper articles (Haruno,1995). The case frames produced by their methods consist of semantic markers in a given thesaurus instead of words. For example, the case frame for the verb "tsumu" is represented as:

[person]ga [vehicle]ni [thing]wo tsumu

[person]ga [mind]wo tsumu

In this kind of case frames, each case is represented by a semantic marker in a given thesaurus such as *[person]* and *[vehicle]*. However they only built case frames for a small set of target verbs and it is difficult for extention.

Large-scale Japanese case frames have been automatically constructed in recent research (Kawahara and Kurohashi, 2006). They first built a large-scale raw corpus from the Web and applied parsing. To avoid the bad effect of parsing error, they made use of Japanese-specific rules to extract reliable predicate-argument structures from the automatic parses. To address the problem of verb sense ambiguity, they finally applied a

clustering process to acquire wide-coverage case frames with different usages for each verb. Their case frames consist of word examples but not semantic features. Example for the same verb *tsumu* is shown below:

(jugyoin)ga (kuruma,hikoki)ni (nimotsu,busshi)wo tsumu (1)

(sensyu)ga (keiken)wo tsumu (2)

where the first case frame for the verb *tsumu* means " load", which can be represented by the example such as "jugyouin ga hikoki ni nimotsu wo tsumu"(workers load cargo on the plane). The second one means "accumulate" which has the expression such as "sensyu ga keiken wo tsumu" (player accumulates experience).

3.3 Framework of Automatic Case Frame Construction

In order to acquire large-scale case frames in other languages such as English and Chinese. Because they do not have the same case marker in Japanese such "が", which often modify a subject case and "を", which is usually a symbol for object case. We make use of large-scale raw corpora instead of annotated resource. In this section, we introduce the specification of our case frames and explain the outline of our construction framework.

3.3.1 Specification of Case Frames

Examples of our case frames are shown in table 3.1. As we mainly focus on verbs and hope to address the verb sense ambiguity problem, every verb has its own several groups of case frames. These case frames of one certain verb are separated according to the verb's usages. In table 3.1, we show two case frames of the English verb "run".

In the first cluster where the verb "run" means "operating" a computer or a server. In the second cluster, it stands for the normal meaning of "run", that is the common action of human or animals, etc. The first column of the table is the verb and the following number in the bracket shows the cluster number. The second column of each case frame is composed of case slots which are expressed by surface cases. In each surface case, "sbj" means subject of the verb, "obj" means object, "objx" is the indirect object and "pp" stands for a prepositional phrase. The following numbers after each instance in the third column are the frequencies they appear in the corpus.

verb	case slot	instance
run(1)	sbj	user:107 computer:79 process:72 ...
	obj	server:6082 programme:5085
	time	year:40 week:16 time:8 today:4
	pp:without	statement:730
	pp:on	network:76 <num>:51 basis:8 ...
	...	...
run(2)	sbj	i:8695 it:6051 he:3672 you:3334
	pp:via	list:1968
	pp:across	them:1976
	pp:into	problem:926 trouble:627 limitation:397
	...	...
...		

Table 3.1: Specification of case frames

3.3.2 Outline of Our Framework

For automatic acquisition of case frames with high quality, tagged corpora which contain many types of annotations can always provide useful information and be convenient for extraction. However, syntactically analyzed corpora with multi-domains are generally too small. In addition, large-scale annotated corpora are very difficult to acquire and too expensive to construct. Many related studies which make use of tagged corpora are always limited by the low coverage or the ability for extension. We consider that large-scale raw corpus with knowledges in multiple domains such as Web is easy to acquire and its large amount of information provide us enough space to create various rules for filtering in order to acquire useful information, and make it feasible to extract case frames in equivalent quality but with much wider coverage range. Figure 3.1 shows the pipline of construction.

Because predicates always convey most of information in natural language, we mainly construct case frame for predicates. In our proposed framework, in the first place, we apply a series of preprocesses to a raw corpus such as segmentation for Chinese, part of speech (POS) tagging to assgin word categories, and chunking which group the words

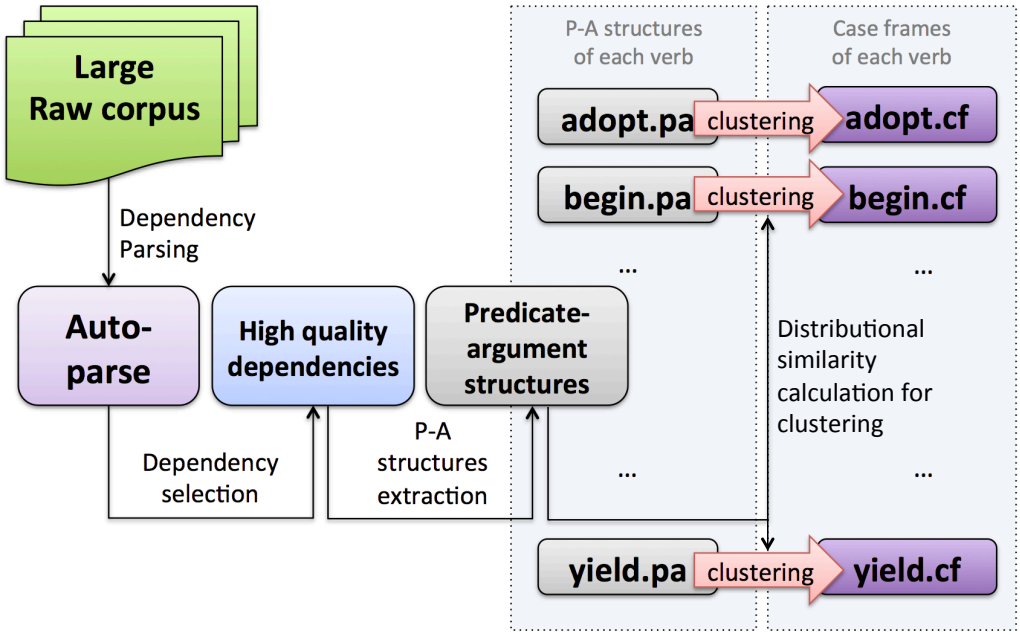

Figure 3.1: Overview of case frame construction

into functional phrase. These preprocesses provide preliminary analysis of raw data for further parsing and convenient for filtering. A dependency parsing step is induced to identify the dependency relation between each component in one sentence especially the relation with verbal phrases.

As we mainly focus on the predicate, with the consideration of that, phrases near the main predicate always have dependency relations and should be acquired as the arguments in case frames. To filter out unrelated components, we create a set of filtering rules which include some common language independent rules and several language specific ones. Then, after extracting a large amount of predicate-arguments from parsed data, we split them by verb into different group where each group only has one common verb. Subsequently, we apply clustering to group the proximate predicate-arguments together to construct the final case frames.

We will explain the construction method in five steps: preprocessing, dependency parsing, filtering out unreliable chunks by language-specific rules, extracting predicate-argument structures, semantic clustering.

3.4 Application to English and Chinese

3.4.1 Preprocessing of Raw Corpus

Before applying deep parsing, shallow parsing processes such as segmentation for Chinese, tagging and chunking are applied to produce essential annotation for deep parsing. In this section, we will describe each preprocess in detail.

Segmentation

Many Asian languages such as Chinese and Japanese are different from English and other western languages. They do not delimit words by white-space. Although a text may be thought of as a corresponding sequence of words, there is considerable ambiguity in the placement of boundaries. For example, in Chinese, many characters can stand alone as words in themselves, while on other contexts the same character becomes the first or second character of a two-character word, and on others it participates as a component of a three-or more-character word. This phenomenon causes ambiguities in Chinese text.

下雨天地面积水
下雨 (raining) 天地 (universe) 面积(area) 水 (water)
下雨天 (rainy day) 地面 (ground) 积水 (flooded)

Table 3.2: Examples of segmentation ambiguities

In the example shown above, the text in the first line is an unsegmentated sentence and it can either be seen as the sentence in the second line which means “I don’t want the bread and give it to him.” and the sentence in the third line which as exactly the opposite meaning “Don’t give him the bread”. In different occasions, there are even more options.

Interpreting a text as a sequence of words is beneficial for many other task in information retreival. Therefore, for Chinese case frames, segmentation becomes the initial task before other processes such as chunking and parsing etc. The performance of segmentation always has a direct effect to the following task. Even though there have been many studies on Chinese word segmentation, it is still a complicated task to find word boundaries without a standard definition of word boundaries in Chinese. There are mainly two types of approaches. One is a dictionary-based method which requires a predefined

dictionary and hand-generated rules (Wu, 1999). This is limited by the impossibility of building a comprehensive lexicon that includes all possible Chinese words and by lack of robust statistical inference in the hand-generated rules. Another one is statistical method such as machine learning which is more desirable and become the major strategy in recent research of Chinese segmentation.

In the machine learning approach for Chinese segmentation, the segmentation problem can be seen as one of the sequence tagging task. For example, the Chinese characters that begin a new word are given the *START* tag, and characters in the middle and at the end of words are given the *NONSTART* tag. So the whole task of segmenting a new unsegmented test sentence becomes a matter of assigning a sequence of tags or labels to the input sequence of Chinese characters.

There are many model proposed to solve this problem such as hierarchical hidden Markov model which incorporate lexical knowledge (Zhang et al. 2003) and maximum entropy models to classify Chinese characters into four tags (Xue 2003). Conditional random fields (CRF) are one popular model that can address many problems such as the difficulty in incorporating domain knowledge effectively into segmentation (Fuchun Peng et al. 2004).

Although Chinese word segmentation locates in the very first beginning of the pipeline in many framework for other applications. However, sometimes, word segmentation is combined with POS tagging or even dependency parsing to be a joint model. In this research, we employ multilingual morphological analyzer (MMA) which is a morphological analyzier that performs word segmentation and POS tagging simultaneously (Kruengkrai, 2009). MMA uses a word-character hybrid model for representing the search space and Margin Infused Relaxed Algorithm (MIRA) for learning the model. MMA can handle unknown words detection task which is another important issue in word segmentation task. The training speed is much faster based on MIRA which is an online learning algorithm.

Part of Speech Tagging

The task of POS tagging is to mark up the words in a text with their corresponding word category such as noun or verb and some additional feature information occasionally such as singular or plural. Tagging is also an important research topic in NLP. Tagging

is always the preprocessing in many NLP applications such as information extraction, question answering, dependency parsing etc. The accuracy of tagging inevitably affect the subsequent process.

POS tagging is not as simple as assigning each word a category based on a dictionary or a list, because some words can have more than one part of speech in different context. For example even the noun "water" can be used as a verb in the context like "I water the plants." There are mainly two kind of information that can be helpful to overcome the difficulty. First, some tag sequences are more likely to be appear than others. For example,the sequence of tags *AT JJ NN* which can stand for the phrase "an old desk" is much more common than the sequence tags *AT JJ VBZ*. Secondly, even one word can has multiple possible tags, but some of them are major and more likely than others. For example "dog" is more often a noun than a verb although we can use "dog" as a verb in some rare case. In the wide used machine learning way to solve tagging task, the problem becomes that, given a sequence $x_{1:N}$ and estimate the most possible output put of tag sequence $z_{1:N}$.

Researchers in early age use a classic model called hidden Markov model (HMM) to solve this problem. The parameters of an HMM is $\Theta = \{\pi, \phi, A\}$ where π stands for the initial state distribution, A is a matrix of transition probabilities. This model is actually a linear chain on hidden state $z_{1:N}$ and observed word $x_{1:N}$. And the tagging problem can be modeled as the joint probability of a sequence of words and a sequence of tags by given the parameters.

For the training of parameters Θ by using corpus with gold standard tagged texts, maximum likelihood estimate (MLE) is usually used. The MLE will find the optimal parameters Θ which maximize the likelihood of observed data. However, using HMM for tagging has its shortcomings. First, HMM models direct dependence between each state and only its corresponding observation. But POS tags may depend not on just a single word bu also other features of the whole sentence. HMM only learns a joint probability of states and observations $p(z_{1:N}, x_{1:N})$, but in fact, we need the conditional probability $p(z_{1:N}|x_{1:N})$ actually.

As a result, many other sequential classification approaches are proposed in later studies. One famous model is the maximum entropy markov model (MEMM) which directly learns with predictive function. However, it always suffer from label bias prob-

lem, but which can be solved by models such as conditional random fields (CRF). In our research, we use Tsuruoka's tagger (Tsuruoka et al., 2005). This tagger makes use of two advantages of sequential classification approaches. One is the efficiency of training CRF needs to perform dynamic programming over the whole sentence in each iteration of optimization. The other advantage is the ability to incorporate other state-of-the-art machine learning algorithm into local classifier. In order to enrich the information that the local classifier can use, Tsuruoka's tagger uses a bidirectional inference algorithm to be an extension of MEMM. This tagger gives conparable performance as state-of-te-art learning algorithms.

Chunking

Chunking process is known as a very popular shallow parsing or light parsing. It is used as a useful preliminary step to deep parsing. This process divides sentences into labeled and non-overlapping sequences of words in order to identify the constituents such as noun groups or verb groups etc. However the information such as internal structures or the syntactic roles are hard to specified. This parsing concept was first proposed by Abney (1991). He was inspired by the psychological studies of Gee and Grosjean (1983) in which psychological evidence for the existence of chunks was found in these studies.

As shown in the example, we induce IOB tags to indicate the boundary of each chunk. IOB tags include an initial mark with a subcategory of POS and basically include two kinds of information. First, IOB tags use the initial marks to grouping constituens to be chunks where B stands for the beginning of a chunk, I means internal part of chunk and O means outside part of chunk. A tag begins with B and numbers of tags begins with I, are normally seen as a chunk, and tags begins with O are always punctuation or end of a sentences. Second, IOB tags can classify these chunks into some grammatical classes with the subcategory of POS. For example a noun phrase (NP) can be considered as two kinds of chunk, B-NP or I-NP. Normally we use around 22 types of IOB tags.

As a result, similar to Chinese segmentation problem, chunking process also can be view as a task to assign IOB taggers to each words. Chunking task was firstly regarded as a sequence tagging problem and solved by using a transformation-based machine learning method (Ramshaw and Marcus, 1995) .

But the input of a chunking process is always not only the raw text but also the POS

word	POS tag	IOB tag
computer	NN	B-NP
can	MD	B-VP
run	VB	I-VP
DNS	NN	B-NP
server	NN	I-NP

tags. Since the task is about to estimation from the features in the surrounding context, machine learning approach seem to be the most appropriate method to solve this kind of problem. Conventional machine learning approach that normally used in POS tagging or segmentation such as HMM or ME usually require a careful feature selection in order to achieve high performance. However these models do not provide a automatic feature sets selection method.

In this paper, we use YamCha (Kudo and Matsumoto, 2000), which is a generic, customizable, and open source text chunker to solve text chunking problem. Yamcha is based on a state-of-the-art learning technique called support vector machine (SVM). SVM is a classifier which uses a strategy to maximize the margin between critical samples and a separating hyperplane. SVM has good performance even with training data of very high dimension. So it can carry out the training considering combinations of more than one feature. Although basically, SVM is a binary classifier, it is needed to extend SVM to a multi-class classifier in order to classify all types of IOB tags. There are mainly two approaches to extend a binary classification task to K-class classification. The first approach is a "one class vs. all others" strategy, which separates one class from all other classes. So if there are totally K types of tags, it is needed to built K classifiers. The second idea is known as the *pairwise classification* which built $K \times (K-1)/2$ classifiers considering all pairs of classes. When given a new data, majority voting is used to decide the final class. YamCha is using the pairwise classification because the amount of training data is less than the method separating one class with all others.

3.4.2 Dependency Parsing

To describe sentence structure in natural language, there are mainly two ways. First one is by dividing the whole sentence into functional phrases or constituents. Constituency grammar was passed through formal logic to linguists like chomsky and used as the basis of formal language theory. Another method is by drawing dependency links connecting each individual words in the sentence. As the computer implementations of dependency grammar have attracted interest for at least 40 years, this kind of parsing concept has become most popular and has been used in many other applications such as machine transaltion, question answering and case frame construction.

When two words are linked with dependency relation, one of then is called head and the other is the dependent. Generally, the dependent is a modifier or complement of the head and the head always play more important role in determining the behavior. Normally, a dependency link between head and dependent is represented by an arrow that point dependent from head. Those arrows compose the dependency tree which is a directed acyclic graph. Dependency tree is wide used to represent the dependency structure in a sentence. In a dependency tree, head are always the present of its dependents. One head may have several dependents as its children but one dependent can have only one direct present. There is always a root of the tree which is usually the main verb in the sentence.

For dependency approaches, there is grammar-based parsing which is based on analytic formal grammars. Different from grammar-based parsing, data-driven approaches were proposed which learn to produce dependency graphs for sentences from an annotated corpus. Comparing to grammar-based model, data-driven model are easily ported to any domain or language as long as there exist respective annotated language resources. There are two dominant models for data-driven parsing. Graph-based parsing (McDonald et al., 2006) and Transition-based parsing (Nivre et al., 2006).

In our research, we employ an MSTparser (McDonald et al., 2005) for English dependency parsing. MSTparser is a graph-based parser that searches for maximum spanning trees over directed graphs. Models of dependency structure are based on large-margin discriminative training methods. We use an extented version of MSTparser called CNP (Chen et al., 2009) for applying Chinese dependency parser. CNP parser can achieve high accuracy due to the use of the features based on subtrees extracted from auto-parsed

data. Comparing with the graph-based model which usually resolves in a global learning procedure, transition-based model is a local learning framework that makes use of greedy search for local optimal decisions.

id	word	POS tag	dep
1	computer	NN	3
2	can	MD	3
3	run	VB	0
4	DNS	NN	3
5	server	NN	4

In this chapter, in order to produce machine-readable data of dependency relation in a text, we use the Conll format which uses numbers to indicate the dependency. As shown in the example, when given a raw text "computer can run DNS server", the word "run" is the main verb of this sentence. Both of the subject "computer", object "server" are arguments that depend on the main verb. So the number in the "dep" label are both 5 which is exactly the ID of the main verb. Also for the word "DNS" which is a modifier of the object, it depends on "DNS", which has the ID of 4.

3.4.3 Filtering of Automatic Parses

There always exist some sentences that are incomplete in grammar or syntax. Those kind of sentence will bring many noises into case frame construction and could not be used to correctly provide the usages of predicates. Similarly, some chunks in a sentence with complicated syntax relations have no dependency relations with the main predicate and should not be acquired for case frame construction. In turn, we define small sets of linguistic rules to filter out unreliable sentences and parses.

Language Independent Rules

Even though different languages have their own grammar and linguistic characteristics, there are usually some common features that can be used to make common filtering rules. For both English and Chinese, we apply some language independent rules to filter out the unreliable sentences and chunks unreliable sentence:

- a sentence which begins with PP
- a sentence which ends with a question mark
- a sentence which includes sign
- a sentence with no VP

unreliable chunks:

- chunks don't have dependency relation with predicate
- chunks with ADJ or ADV

We mainly construct case frames for predicates, and sentences with no predicates can not be used in construction. Also, some special punctuations, such as a question mark or a colon, always change the sentences into different form or even change the phrase order. We directly filter out all the sentence with those kinds of punctuations.

Since different languages have their own features, we also create language-specific rules to filter out unreliable sentences and chunks in each language.

Rules for English

We discarded sentences and chunks by using the following rules for English:

unreliable sentences:

- a sentence which begins with VP
- a sentence in which comma is next to VP
- a sentence with no NP before VP
- a sentence where VP appears as Infinitive or participle

unreliable chunks:

- all the chunks after comma
- all the chunks after WH phrases
- all the chunks after after second VP

By applying this set of rules, we avoid some complex cases which contain several verbs, especially some clauses led by *WH* which involve more complex dependency relations. For example, in the sentence of *He went back where he came from,* the *where* clause was discarded by the rule in order to extract a simple structure.

Rules for Chinese

The rules for Chinese are as follows: unreliable sentences:

- a sentence ends with DE
- a sentence which contains a DE directly after VP
- a sentence which matches the pattern of VP NP DE NP

unreliable chunks:

- all the chunks before the last comma
- all the chunks before the second VP from bottom
- all the chunks after the third VP form top in a BA phrase

These language rules properly fit Chinese linguistic characteristics.Some special case in Chinese such as sentences end with character "的" are always stands for emphasizing and have different phrase order. Different from English, Chinese nearly does not have any marker words for clause such as where or which etc. For example in the sentence "我 (I) 希望 (hope) 他 (he) 早日康復 (recover soon)", there would be usually a clause marker which changes this sentence into "I hope that he can recover soon". In order to reduce the complexity and improve the accuracy, we only consider one predicate and retain the first one in English case. However, we retain the second verb and find that the arguments around the second predicate can always compose a complete sentence.

3.4.4 Extraction of Predicate-argument Structures

PP-attachment Disambiguation

Syntactic ambiguity problem can effect the precision of predicate-argument structures extraction. There is a syntactic ambiguity problem in English, which is known as the PP-attachment problem which always occurs in the sentences contain prepositional phrases.

When a sentence has the pattern of *VP NP1 PP NP2*, the *PP* can be either modifying the *VP* or the *NP1*. For example, in the sentence *VP:hit NP1:ball PP:with NP2:bat*. The prepositional phrase "with bat" is likely to modify the verb "hit" to describe how the ball is hit. In the sentence *VP:see NP1:dop PP:on NP2:road*, the prepositional phrase "on road" attaches to the *NP1 dog* with higher probablility, it is more likely to describe the dog's location.

When we extract predicate-argument structure, the prepositional phrases which do not attach to the main predicates will become noises in the construction because they nearly do not provide information about the predicates. In order to automatically distinguish whether the prepositional phrase is related to the verb in a sentence, a classifier is needed in our framework. Since there are only two options, this problem can be viewed as a binary classification problem (Kawahara and Kurohashi, 2005) and we employed SVM to solve this problem.

We acquire training set both tagged corpus. The annotated data in Penn Treebank is used to construct the correct learning data for PP-attachment problem by IBM research group. Further more, We ultilize the same corpus which is used in case frame construction to extract precise unambiguous examples. There are two types of unambiguous examples:

- *He hit vp it with a bat*
- *The bench in the park is broken*

In the first sentence, the prepositional phrase "with a bat" is a modifier of the predicate "hit" because one prepositional phrase is impossible to attached to a pronoun. The prepositional phrase in the second sentence must be a attachment of the the noun "bench" because there are no other possible heads existing. We extract unambiguous examples based on the above two heuristics that one prepositional phrase can not attach to a pronoun and must attach to the direct preceding noun. In turn, following types of examples are extracted:

- triple of $(VP, PP, NP2)$ from the sentence where a verb is followed by a pronoun and a prepositional phrase
- triple of $(NP1, PP, NP2)$ from the sentence where a noun phrase is followd by a prepositional phrase

Using the above extracted information, we further calculate pointwise mutual information between *VP* and the combination of *PP_NP2* as:

$$I(VP, PP\_NP2) = log \frac{\frac{f(VP, PP_N P2)}{N}}{\frac{f(VP)}{N} \frac{f(PP_N P2)}{N}}$$

The pointwise mutual information between *NP1* and the combination of *PP_NP2* is calculated in the same way:

$$I(VP, PP\_NP2) = log \frac{\frac{f(NP1, PP_N P2)}{N}}{\frac{f(NP1)}{N} \frac{f(PP_N P2)}{N}}$$

We also take these kinds of mutual information into consideration as a set of important features in the classification. The input is a quadruple $(VP, NP1, PP, NP2)$ and our task is to classify it as VP or $NP1$. The resulting class VP means the preposition phrase is modifying the verb. The class NP1 means that the prepositional phrase is attaching to the noun $NP1$ and as a result should be discarded as a noise.

Transformation Rules

From the reliable chunks, predicate-argument structures are extracted in a straightforward way. We simple create a set of transformation rules to transform parses to reliable predicate-argument structures.

- convert *VP* to *pred*
- convert *PP* and the following *NP* into prepositional phrase
- convert *NP* preceding the predicate to *sbj*
- convert *NP* following the predicate to *obj*
- oonvert the left *NP* to *obj2*
- we change *BA* and *LB* to *ba*[1] in Chinese

[1]As for the BA phrase, it is a special case of grammar in Chinese which puts the object behind the verb and change the phrase order from original SVO to SOV, but the meaning is almost the same. e.g., the sentence 我 *(I)* 把 *(BA)* 面包 *(bread)* 吃了 *(ate)* as the same meaning with 我 *(I)* 吃了 *(ate)* 面包 *(bread).*

For example, one predicate argument structure can be converted to *sbj:I pred:put obj:pen pp:in:pocket*.For English prepositional phrase, there is always a common pattern which contain a prepositional in front of an argument. For example, an English prepositional phrase can be written as *PP:on NP:desk*. We simply convert them into the format of *pp:on:dest*.

However, there are several patterns in Chinese prepositional phrase. One common pattern is the same with English such as *PP:*在 *NP:*中国 (in China) which we simple change the form to *pp:*在*:*中国. Another pattern is that, the noun argument is located between two prepositional words. For example, *PP:*在 *NP:*村子 *LCP:*里 (inside the village). We change the form of this pattern to *pp:*在... 里*:*村子. Also in some cases, the preposition is omitted in a prepositional phrase such as *VP:*放 *NP:*車 *LCP:*里 (put in the car). We change this pattern of prepositioanl phrase into *lcp:*里*:*車

3.4.5 Clustering of Predicate-argument Structures

After the extraction of predicate-argument structures, we cluster the predicate-argument structures into their usages. For each predicate-argument structure, we firstly choose one important argument which can mostly indicate the predicate's meaning and is for most of the time the object. We call it "key argument". We initially group the instances with the same key argument to initial clusters. To implement further clustering of all the initial clusters, we utilize the method of similarity calculation similar to Japanese (Kawahara and Kurohashi, 2006) which considers two aspects of initial clusters: the similarity of case slot patterns and the similarity between each instance in the same case slot. To calculate the similarity between two words, we use distributional similarity, which is based on the hypothesis that words with similar semantic features always share the similar contexts (Hindle, 1990). In Hindle's research, he described a method of determining the similarity of nouns on the basis of a metric derived from the distribution of subject, verb and object in a large text corpus, which is a purely syntax-based similarity measurement. We ultilize our extracted predicate-argument structures as the corpus to measure the similarities between nouns.

As the example shown in figure 3.2. C_1 and C_2 stand for the initial clusters. Each initial cluster is the collection of predicate-argument structures sharing the same key argument which are *ship* as an object in C_1 and *plan* as an object in C_2. Each case slot are

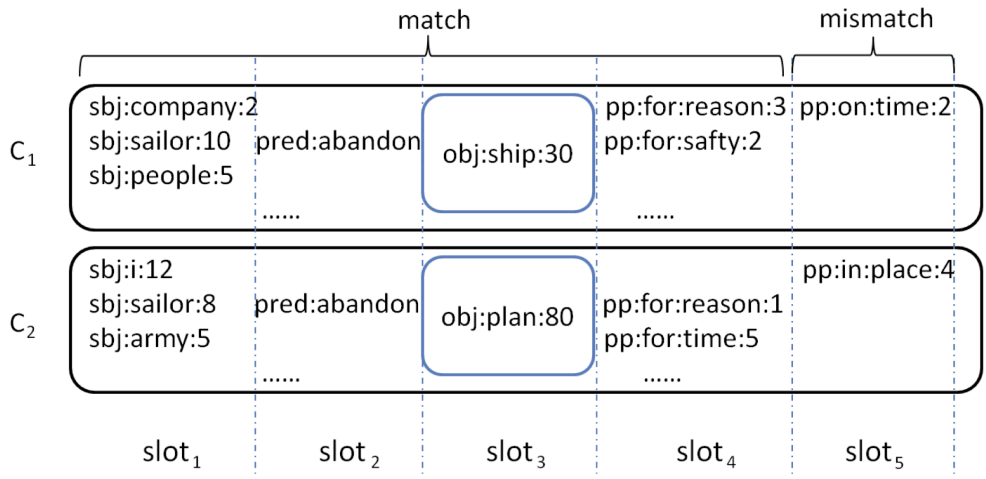

Figure 3.2: Initial cluster

represented by its the instances. In the final clustering process, we want to calculate the similarity between each initial cluster and merge the similar initial clusters into one final cluster. First, we compare all the instance in each matched case slot pair. In this example, $slot_1$ to $slot_4$ are matched and $slot_5$ are mismatched for each initial cluster. Then we calculate the case similarity between each matched case slot based on the average frequency of most similar instances from each pair. We define $slot_i$ for C_1 as S_{1i} and for C_2 as S_{2i}. The case similarity between S_{1i} and S_{2i}, $CaseSim(S_{1i}, S_{2i})$ is calculated as follows where e_1 stands for the instances in S_{1i}, e_2 means the of all the instances in S_{2i}, and $sim(e_1, e_2)$ is the similarity between two instances.

$$CaseSim(S_{1i}, S_{2i}) = \frac{\sum_{e_1 \in S_{1i}} |e_1| \cdot max\{sim(e_1, e_2) | e_2 \in S_{2i}\} + \sum_{e_2 \in S_{2i}} |e_2| \cdot max\{sim(e_1, e_2) | e_2 \in S_{2i}\}}{\sum_{e_1 \in S_{1i}} |e_1| + \sum_{e_2 \in S_{2i}} |e_2|}$$

Considering that, slots with more instances play more important roles, we add weight to each $CaseSim(S_{1i}, S_{2i})$ and for two initial clusters C_1 and C_2, we define a weited similarity $WeightedCaseSim(S_{1i}, S_{2i})$ which is calculated as the following formula. Each weight is calculated based on the number of instances in each case slot.

$instance_i \in role$

$$vector(role) = \frac{\sum_{i=1}^{n} vector(instance_i) \cdot freq(instance_i)}{\sum_{i=1}^{n} freq(instance_i)}$$

$$WeightedCaseSim(C_1, C_2) = \frac{\sum_{i=1}^{m} \sqrt{|S_{1i}||S_{2i}|} \cdot CaseSim(S_{1i}, S_{2i})}{\sum_{i=1}^{m} \sqrt{|S_{1i}||S_{2i}|}}$$

where $|e_1|$ and $|e_2|$ are the frequencies, $|S_{1i}|, |S_{2i}|$ are total numbers of instances in each case slot. From $S_{11}toS_{1m}$ and from $S_{21}toS_{2m}$ is the matched case slots.

As another aspect of similarity between two initial clusters, we measure the matching level $Alignment(C_1, C_2)$ between them. Basically it is calculated as the percentage of instance number in matched case slots. When given the total number of case slots for each initial cluster p, q, the alignment matching level of two initial clusters is measured as:

$$Alignment(C_1, C_2) = \sqrt{\frac{\sum_{i=1}^{m} |S_{1i}|}{\sum_{i=1}^{p} |S_{1i}|} \times \frac{\sum_{i=1}^{m} |S_{2i}|}{\sum_{i=1}^{q} |S_{2i}|}}$$

We combine these two aspects with reflect the similarity to be the similarity between two initial clusters.

$$Sim(C_1, C_2) = WeightedCaseSim(C_1, C_2) \times Alignment(C_1, C_2)$$

When applying the final clustering for all the initial clusters with similarity of $Sim(C_i, C_j)$, there exist several clustering method such as nearest neighbor method, furthest neighbor method or group average method. For example, in the process of clustering, if there is a large cluster which combine several similar initial clusters together, when a new initial cluster comes, we decide whether merge it to the large cluster.

We take figure 3.3 as an Example of the final clustering process. Initial cluster C_1 to C_4 are already clustered as a big cluster. Subsequently, we consider the initial cluster C_5 and decide whether to cluster C_5 into the big cluster. In nearest neighbor method, we compare the new coming initial cluster with all the inital clusters in the big cluster with $Sim(C_5, C_i), C_i \in bigcluster$. Then take the biggest similarity as the similarity between new coming initial cluster and the big cluster.

$$Sim(C_5, big\ cluster) = maxSim(C_5, C_i), C_i \in big\ cluster$$

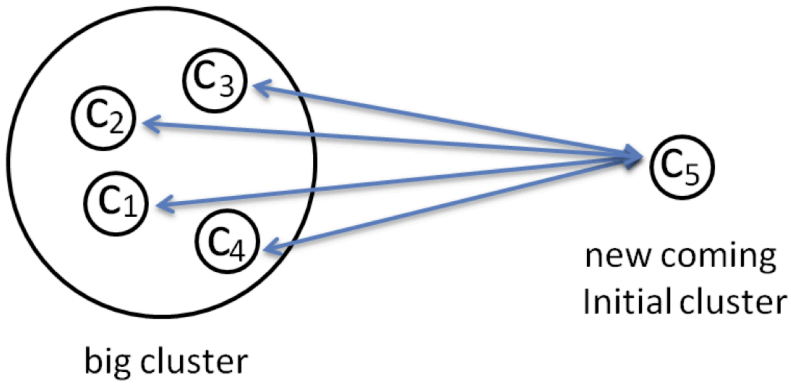

Figure 3.3: Example of clustering

So one we find similarity larger than given threshold, we simplely stop comparing and group the new coming initial cluster into the big cluster. In furthest neighbor method, we take the smallest similarity as the similarity between new coming initial cluster and the big cluster.

$$Sim(C_5, big\ cluster) = minSim(C_5, C_i), C_i \in big\ cluster$$

In other words, unless every component in the big cluster has the similarity with new coming initial cluster larger than threshold, we do not merge them into one cluster. The group average method considers the average value of all the similarity and then use the average similarity to decide whether to merge the new coming initial cluster.

$$Sim(C_5, big\ cluster) = minSim(C_5, C_i)/N, C_i \in big\ cluster$$

The nearest neighbor method is a simple algorithm but always lead to the problem of imbalance. That is, for one predicate, some case frames have much more instances. For example, if the word “car” and “plane” has similarity of 0.6 and clustered as a big cluster. Another word “line” has similarity of 0.7 with “plane” but 0.3 with “car”, the imbalance result will group all the three words into one cluster even though they are not all similar to each others. Although The group average method can solve this problem, it increases computational complexity and always takes much more time for clustering. The furthest neighbor method usually performs well in grouping balance and computing time.

3.5 Experiments

3.5.1 Constructed Case Frames for English and Chinese

For English case frames we made use of 200 million sentences extracted from the Web and for Chinese we utilized Center for Chinese Linguistic (CCL) corpus for construction, which consists of 10 million sentences including abundant domains of words varying from history, literature, magazine, translation to movie, drama, internet and speaking language. For the process of segmentation of Chinese, a MIRA-based system MMA was used which achieves accuracy of 94%. For English POS tagging, we use Tsuruoka's English POS tagger which can perform precision over 97%. For chunking, we exploit Yamcha, which is based on SVM, and utilized Penn Treebank and Chinese Treebank (CTB) to train a chunker for each language. Yamcha reached the precision of over 95% for both language. We used MSTparser to apply English dependency parsing which can achieve the precision about 92%. For Chinese, we employed CNP which is an extension of MSTparser and has the precision of 88%.

We extracted predicate-argument structrues for about 14,000 English predicates and 13,000 Chinese predicates and implemented clustering to complete case frame construction for each predicate. We show the example of case frame both for Chinese and English. Finally, we proposed several method to evaluate the performance of our experimental result.

3.5.2 Evaluation

To evaluate the case frames we acquired from large-scale raw corpora, we mainly apply the evaluation in two ways. First we want to know how reliable the predicate-argument structures are produced by the proposed framework, and we use gold standard data to do an automatic evaluation on predicate-argument structures. Secondly to see the efficiency of verb sense disambiguation, we manully evaluate on the clustered case frame and compare with subcategorization frames which is used as the baseline.

verb	surface case	instance with frequency in original corpus
visit(1)	sbj	people:86, tourist:70, student:60 ...
	obj	[name]:14736
	pp:with	[name]:25, family:15, friend:8 ...
	pp:on	occasion:28, day:13, way:5, trip:5 ...
	pp:in	july:39, hospital:26, sprint:16 ...
	...	...
visit(2)	sbj	host:508, user:445, consumer:281 ...
	obj	site:11321, page:3771, website:2619 ...
	pp:on	basis:36, [num]:8, web:4 ...
	sbar	if:117, because:93, before:76, as:43 ...
	pp:at	time:13, university:4, school:4 ...
	...	...
...		

Table 3.3: Examples of English case frames

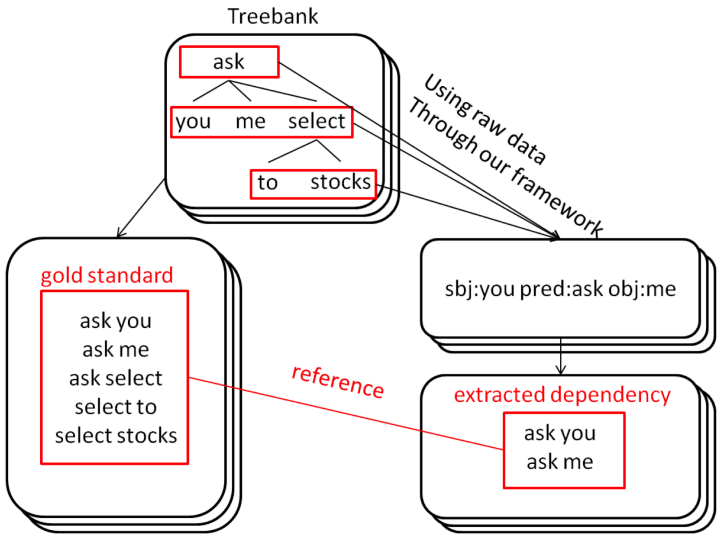

Figure 3.4: Automatic evaluation of predicate-argument structures

verb	surface case	instance with frequency in original corpus
打开(1)	sbj	产品:12, 信息:3, 限制:1, 商品:1, 国家:1 ...
	obj	市场:65, 销路:27
	pp:给	企业:1
	pp:在.. 上	市场:2
	pp:在	市场:3, 海外:2, 国:1, 中国:1, 独 联体:1 ...
	...	...
打开(2)	sbj	他:17, 太太:4, 人:3, 自己:3, 演 员:2 ...
	obj	门:237, 大门:104, 车门:36, 房门:31 ...
	pp:给	他:2, 她:2, 你:1 我们:1, 应用:1, 罪 恶:1 ...
	pp:用	钥匙:2, 她:1, 金钱:1, 手段:1 ...
	pp:向	妇女:1, 世界:1, 学者:1, 产品:1 ...
	...	...
...		

Table 3.4: Examples of Chinese case frames

Automatic evaluation of predicate-argument structures

In order to evaluate the filtering rules that we used to extract highly-reliable predicate-argument structures, we automatically evaluated them using PTB for English and CTB 5.0 for Chinese. We use file wsj_23 to test for English and file chtb_271-300 in CTB as testing data for Chinese. First, we applied our framework to the raw texts in the treebank, and extracted predicate-argument structures. In each predicate-argument structure we extracted, the dependency relation is defined as: every argument depends on the predicate. Then, we use the existing dependency annotation in each treebank to extract gold standard dependency pairs. We calculate the precision as the percentage of correct dependency pairs among acquired pairs. We calculate the recall as percentage of correct dependency pairs among the total dependency pairs in the treebank.

As we can see in Table 3.5, for both Chinese and English we achieved a precision of over 97% and a recall of around 30%. From our error analysis, most of the incorrect dependency pairs is due to dependency parsing errors. For example, in the Chinese sen-

language	precision	recall
English	0.977	0.357
Chinese	0.982	0.337

Table 3.5: Automatic evaluation of predicate-argument structures

tence “殴打 (assault) 事件 (event)”, the nount “事件” is always parsed as an object of the verb “殴打”. However, there is an omitted character “的” betwwen these two words and the verb “殴打” is actually a modifier of the noun “事件”. This kind of problem can be solved by using the feedback of constructed case frames in the future work.

Manual Evaluation of Case Frames

We built subcategorization frames from the same corpora for each predicate for comparison. For both subcategorization frames and case frames, we conducted two types of evaluation: slot-based and frame-based.

For the slot-based evaluation, we manually judged whether each case slot is well constructed by the following criterion:

- 80% of the instances in the case slot are semantically similar.
- case slots that have only one instance are not counted.

We then calculated the accuracy of both kind of frames by the percentage of good case slots.

As the above evaluation is based on case slots, we also conducted a frame-based evaluation. For each case frame we built, it can be seen as a good frame only if it satisfies the following two conditions:

- above 80% of its key phrases are semantically similar.
- the key phrases in the frame must be semantically independent with any other existing frames.

The second point means that, key phrases in one case frame must not be similar with any others. In subcategorization frames, we simple choos the noun directly after or before

predicate	subcat frames	case frames
visit	0.44 (80/181)	0.67 (16/24)
run	0.20 (40/201)	0.50 (36/72)
begin	0.18 (33/179)	0.52 (42/80)
believe	0.39 (27/70)	0.55 (20/36)
ask	0.28 (43/156)	0.52 (14/27)
find	0.17 (55/332)	0.53 (9/17)
add	0.22 (46/208)	0.52 (14/27)
total	0.27	0.54

Table 3.6: Slot-based evaluation of English case frames

the predicate to be the key phrase. The accuracy is calculated as the percentage of good frames.

The evaluation results for seven frequent English and Chinese verbs are shown in tables 3.6, table 3.9, table 3.8 and table 3.9. As we can see, the clustering method in the construction of case frames merged most similar instances in each case slot, so the total number of case frames is much smaller than subcategorization frames, and case frames outperformed the subcategorization frames obviously, because subacategorization frames is only clustered by syntactic patterns without considering the semantic aspects.

3.5.3 Discussion

In fnial clustering, Chinese case frames perform worse than English case frames by about 10% in general. In Chinese case frames, the experimental result shows that there are some inaccuracy in word similarity and lead to strange clustering result. For example, initial clusters with key phrases such as “教育 (education)” and “服务 (service)” are clustered together with high similarity. This problem due to the small size of Chinese corpus we used as extracting the predicate-argument structures. The distributional similarity calculated from the predicate-argument structures would become less accurate and make direct effect to the final clustering. We are planing to use larger corpus such as Web corpus for Chinese case frame construction.

predicate	subcat frames	case frames
产生	0.37 (13/35)	0.54 (15/28)
发表	0.35 (59/111)	0.59 (32/54)
发展	0.26 (40/154)	0.54 (26/48)
贡献	0.22 (4/18)	0.71 (5/7)
进入	0.25 (28/111)	0.61 (9/17)
开展	0.28 (31/110)	0.51 (31/60)
提出	0.29 (40/141)	0.57 (60/105)
total	0.29	0.58

Table 3.7: Slot-based evaluation of Chinese case frames

predicate	subcat frames	case frames
visit	0.24 (8/33)	0.75 (3/4)
run	0.05 (2/38)	0.67 (4/6)
begin	0.15 (5/34)	0.73 (8/11)
believe	0.28 (5/18)	0.80 (3/5)
ask	0.39 (13/33)	1.00 (2/2)
find	0.30 (18/60)	0.75 (6/8)
add	0.18 (7/39)	0.78 (7/9)
total	0.23	0.78

Table 3.8: Frame-based evaluation of English case frames

For the construction of all the predicates, we set a common threshold for the similarity between initial clusters when clustering them into final case frames. That is, all the initial clusters with similarity larger than threshold will be clustered for all the predicates. This leads to the problem that, case frame for some predicates are over clustered and some are too sparse. For example,“i”,”you”,”file” are clustered in one case frame in some cases even though “file” does not similar to other two words. Also key phrases like “page” and “site” are not clustered in some case frames.

predicate	subcat frames	case frames
产生	0.11 (8/33)	0.70 (7/10)
发表	0.21 (2/38)	0.50 (8/16)
发展	0.09 (5/34)	0.71 (15/21)
贡献	0.25 (5/18)	0.67 (2/3)
进入	0.10 (13/33)	0.71 (15/21)
开展	0.10 (18/60)	0.73 (11/15)
提出	0.21 (7/39)	0.67 (13/20)
total	0.15	0.66

Table 3.9: Frame-based evaluation of Chinese case frames

Many linguistic variations cause parsing errors and lead to the decrease of accuracy in the acquisition of predicate-argument structures and thereby cause the bad effect to the final clustering. We can solve the following problems by using the feedback information from the case frames constructed in the first stage.

In Chinese, in the sentences with the pattern of *VP NP1 DE NP2*, for example in the sentence of *VP(*制定 *set) NP1(*计画 *plan) DE (*的*) NP2(*时间*)*, the whole phrase of *NP1 DE NP2* can be seen as the object of the *VP*, so the sentence means *set the time of plan.* Also the part contains *VP NP1* can be seen as a modifier of *NP2* and the sentence's meaning is going to be *the time of setting plan*, which is nothing but a noun phrase and should be discarded.

We make use of the case frame built in the first stage to calculate the combination frequency of *VP NP1* and *VP NP2*. If *VP NP1* are the most common pattern, then we can judge the parsing result as the second one we mentioned before, and vice versa.

Also, omission is one common feature of human language. For instance, many clause markers like *which, where* or *that* are missing all the time. This leads to the wrong parsing result of our system. For example in the sentence *I heard (that) the machine exploded*, we can easily incorrectly make *the machine* as the direct object. Similar case in Chinese, in the sentence of "人为(地) 破坏" which means deliberately destroying, the adverb markesr are always omitted and that can even affect the morphological analysis

and make our system to mistakenly see “人为” as the subject.

We ultilize our case frames constructed to see whether a predicate is always leading a complex clause in English, and in order to reconsider its object. For Chinese we make the adverb with character 地 as an important clue to judge whether a subject is actually a adverb or not.

3.6 Summary

In this chapter, we proposed a framework for automatically constructing case frames for multiple languages. We also apply an evaluation method from different aspects. From the evaluation result, we can see that our framework successfully extracted highly-reliable predicate-argument structures from raw corpora and well clustered instances with similar semantic features to produce the final case frames. We found that many parsing errors can be solved by using the feedback of constructed case frames resource. We are planing to implement the feedback strategy and evalutate the efficiency. Also, a larger corpus for Chinese is under construction for improving the construction.

Chapter 4

Dependency Parsing using High-quality Knowledge

This chapter presents an application of Chinese auto-acquired knowledge for dependency parsing. Morphological and lexical information are crucial in dependency parsing. However, it is difficult to learn such information from limited, small-scale, and manually annotated training data. Instead of manually increasing the size of annotated data, we use a large amount of automatically extracted syntactic knowledge to improve the performance of dependency parsing.

4.1 Introduction

To achieve the goal of text understanding, one of the prerequisite fundemantal NLP tasks is parsing or syntactic analysis. Parsing a sentence is to indicate how a sentence is composed by all the words by following language specific grammars. Parsing mainly focuses on resolving the structural ambiguity of a sentence without describing the exact meaning of the sentence. For example, the syntactic ambiguity in the sentence "eat a salad with a fork" can be solved by presenting this sentence as a syntactic tree.

According to different perspectives of syntactic parsing, it can be categorized as two types of parsing techniques. A constituent parser breaks a whole sentence into sub-phrases. A sub-phrase may be further seperated into finer sub-phrases. A constituent tree is composed by non-terminals and terminals. Non-terminals normally represent their

phrase types. Terminals are the actual words in the sentence. An automatic constituent parser must find an optimal constituent structure with the restriction of legal formal grammars.

The other type of syntactic parsing mainly focus on the pair-wise relations between words in a sentence. By abandoning the concepts of non-terminals and terminals, dependency parse trees do not contain phrasal categories. Instead, dependency parsing connects each word with a directed arc according to the syntactic relationships. Each node in the tree structure represents an actual word. Child nodes are words depending on the parent nodes. Formally, a directed arc is pointed from the parent node to its child node. Thus, a parent node is more important than a child node for most of the time. Different from constituent parsing, in a dependency tree, the arcs are sometimes labeled with syntactic roles to indicate the type of relation between a parent and a child. Because dependency parsing is more representative when indicating the relation between words, which is exactly what we are aspiring after, we choose dependency parsing for syntactic analysis.

Despite of the type of syntactic parsing, a data-driven approach always relies on a corpus of training data which contains manually annotated tree structures. All the parameters and information can only be learned from the training data. For example, even a parser can correctly judge the relation between a verb “eat” and its direct object “pair”, in some extreme cases, it is still not able to judge the relation between the verb “eat” and “orange”. Even though this relation is explicit for human judge, a system can fail to analyze correctly just because this pattern has not appeared in the training data. In practice, if one tries to apply such a dependency parser on the data from different domains, the performance will drop inevitably. The size of training data is always deemed as one of the main causes letting parsing techniques hit a bottleneck. Instead of expanding the training data, it is more promising to make use of large-scale knowledge acquired from raw text. We propose a method that makes use of high-quality knowledge extracted from raw text.

The rest of this chapter is organized as follows: Section 4.2 contains some related work for dependency parsing. Section 4.3 describes the approach that makes use of the high-quality knowledge. Seciton 4.4.1 details the experimental settings. Section 4.4.2 shows the experimental results. We also give some discussions in Section 4.5. Finally, we summrize our chapter in Section 4.6.

4.2 Related Work

In dependency parsing, transition-based [47] and graph-based [36] approaches are two exemplary types. They solve dependency parsing by adopting quite different views of the problems. A transition-based dependency parser utilizes a sequence of transition actions such as "Shift" and "Reduce" to link word pairs and to further construct a tree structure. The most appropriate transition action sequence is learned from the training data. A transition-based parser analyzes a sentence from left to right in a linear time, but has slightly low accuracy. A graph-based dependency parser normally creates a complete dependency graph including all the words (in most cases, a pseudo word *ROOT*, which indicates the root node of the dependency tree, is also included). Then the dependency parsing process is viewed as finding the highest scoring tree from the complete graph. Also, some restrictions must be followed to keep the legality of the output dependency tree.

In this study, it is difficult to apply the auto-acquired knowledge on transition-based dependency parser because our knowledge does not has much relationships with the transition actions. On the other hand, the concept of graph-based dependency parsing well conforms to the situation. As a result, we only apply the knowledge on a graph-based dependency parser.

There are several studies that tried to make use of addtional unlabeled data to develop a semi-supervised system in quest of better parsing performance. The main idea is the same as our motivation, which takes the advantage of facility of unlabeled data. To use the unlabeled data in the lexical level, Koo et al. (2008) used the unlabel data for learning word clusters. Word clusters are then used as extra features to compensate the data sparseness problem. Zhou et al. (2011) and Bansal and Kleinand et al. (2011) use word co-occurrence counts from the Web as a lexical-level feature. Another way is to apply dependency parsing on the unlabeled data. After that, one can pick up some high-quality auto-parsed instances and directly add them into the original training data [24, 55, 4]. However, it is difficult to pick up reliable sentences as addtional training data. Using this type of method has gained less improvement for dependency parsers than constituent parsers [33]. The most related approaches are those using partial parse trees of unlabeled data. Van Noord (2007) and Chen et al. (2009) used word pairs or subtrees from the auto-

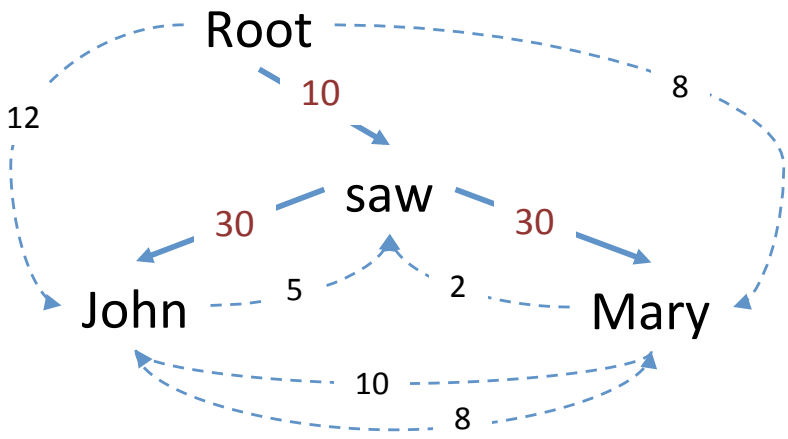

Figure 4.1: Example of graph-based dependency parsing

parses as additional features for dependency parsing. Some studies extracted sub-trees with a certain style using unlabeled data from target domains and used them as additional training data [32] and achieved a few improvements especially in domain adaption. Different from our approach, the abovementioned studies seldom considered the quality of the partial trees from the unlabeled data. Instead, we use high-quality knowledge and investigate the effect by knowledge quality.

4.3 Dependency Parsing using High-quality knowledge

4.3.1 Graph-based Dependency Parsing

In this section, we give a brief introduction of graph-based dependency parsing. Take figure 4.1 as an example. During parsing, a graph-based dependency parser initially connects every word in the sentence with a directed edge. Thus, there are two edges with a different direction between each word pair. Also, to indicate which word is the root node in the dependency tree, it is common to induce a pseudo word “ROOT”. Because in a legal dependency tree, “ROOT” can only be a parent node and is allowed to have only one child node. There is only one directed edge between “ROOT” and the other nodes in the initial complete graph.

In the initial complete graph, there are a large number of possible dependency trees.

A parser tried to find the best dependency tree with the highest score. In formal definition, given an input sentence:

$$X = x_1 x_2 ... x_n,$$

the set of all possible dependency trees over X is defined as $\phi(X)$. Each dependency tree is composed by directed arcs that can be represented by a head node h and its modifier node m. Each node is represented by its word ID according to the word order in the original sentence and the ID of "ROOT" is normally 0. Thus a candidate tree can be defined as:

$$Y = \{(h, m) : 0 \leq h \leq 0, 0 < m \leq n\},$$

where (h, m) stands for a directed arc from the head x_h to the modifier x_m and n is the sentence length. Note that a modifier is always larger than 0 because of the restriction that "ROOT" is illegal to be a modifier node. The best dependency parsing tree is then defined as:

$$Y^* = \arg\max_{Y \in \phi(X)} score(X, Y).$$

To calculate the score of a certain parse tree, the simplest method is the first-order graph-based dependency parsing, which assumes that the dependencies in a tree are independent from each other. Therefore, calculation of the score of a dependency tree $score(X, Y)$ can be in an arc-factorization style:

$$score(X, Y) = \sum_{(h,m) \in Y} score(X, h, m).$$

In Figure 4.1, the best dependency tree which holds the highest is the tree with solid arcs, where "saw" is labeled as the direct child node of "ROOT". "John" and "Mary" are considered to be two child nodes of "saw". For longer sentences, dynamic programming based decoding algorithms [10] are commonly applied to solve the problem for the searching problem when finding the best tree among all the candidates. In a machine learning-based approach, to determine the score of each directed arc between a head and a modifier $score(h, m)$, a feature based representation is always used. The score function for each directed arc can be further factoried as follows:

$$score(h, m) = \mathbf{w} \cdot \mathbf{f}(h, m),$$

$\mathbf{f}(h, m)$ is a feature vector which is extracted from the head and modifier node according to pre-defined features. $\mathbf{w}$ is the a weight vector which has the same dimension with the feature vector $\mathbf{f}(h, m)$. More specifically, because multiple features are disigned for each head-modifier pair in most cases, the score function $score(h, m)$ is always a linear addition of weighted feature funtions, which can be written as:

$$score(h, m) = \sum_{i}^{N} w_i \cdot f_i(h, m),$$

where N is the total number of features for the head-modifier pair. w_i is the weight for the correspond feature funciton f_i, expressing the importance of this feature in a sense. In a supervised approach, each weight w_i is learned from a maually annotated dependency trees such as a treebank.

To improve the performance of a graph-based dependency parser, one can explore features of more non-local subtrees instead of a head-modifier pair. For example, a second-order graph-based dependency parser [35] factorized the score function by sub-trees which contain two directed arcs. Third-order [27] and high-order [52] models use sub-trees with more directed arcs. However, higher-order dependency parsing leads to a huge increase of calculation and thus becomes more time consuming. Instead, we propose to refine the weight vectors for the first-order features by inducing extra high-quality knowledge.

4.3.2 Dependency Selection from Auto-parses

Instead of directly using all the automatic parses, we apply a dependency selection approach and then extract predicate-argument structures from the high-quality dependencies. This idea is based on the fact that a dependency parser tends to analyze different types of text in different level of performance. Take the two sentences “they eat salad with a fork” and “they eat salad with sauce” as examples. These examples have the PP-attachment ambiguity problem, which is one of the most difficult problems in parsing. The two prepositional phrases ‘with a fork’ and ‘with sauce’ depend on the verb ‘eat’ and the noun ‘sauce,’ respectively. However, these two cases can hardly be distinguished by a dependency parser due to the lack of knowledge like case frames. Therefore, we want to judge this kind of structure to be unreliable. Consider another similar sentence “they eat

it with a fork." Since the prepositional phrase 'with a fork' cannot depend on the pronoun 'it' but only on the verb phrase 'eat,' this case can be clearly judged as a highly reliable dependency.

We employ the high-quality dependency selection approach [18], which shows good performance not only in in-domain cases but also out-of-domain cases. This method first trains a base parser using a part of treebank. Then, they apply dependency parsing on the raw text of another part of the same treebank in order to collect training data for dependency selection according to the gold-standard annotations. They use context features and tree-based features, which are thought to affect the selection approach. Then, SVM is employed to solve the binary classification problem that classifies if each dependency is high-quality or not. We do not apply a high-quality parse selection approach [50] because we believe that there still exist many high-quality dependencies even in low quality parses which could be also informative.

4.3.3 Predicate-argument Structure Extraction

Predicate-argument structures mainly capture the syntactic relations between a predicate and its arguments. Building wide-coverage case frames for each verb is basically to apply clustering of predicates-argument structures of each predicate. Japanese predicate-argument structures have been successfully extracted and used for case frame construction [21], where each argument is represented as its case marker in Japanese, such as 'ga', 'wo' and 'ni'. However, for other languages such as English and Chinese, there are no such case markers that can help clarify syntactic relations. Therefore, instead of using case markers like in Japanese, we represent each argument by its syntactic surface case (e.g., subject, object, prepositional phrase). Kawahara and Kurohashi (2010) used a chunking-based approach for large-scale predicate-argument structure acquisition. Instead of capturing dependency relations, this method uses language-specific filtering rules and only selects surrounding arguments, therefore lacks multilinguality.

In order to extract high-quality predicate-argument structures from automatic parses, we define a simple set of extraction rules for each language. First, to reduce the complexity of multi-verb cases, we only use the last predicate in each sentence. We only maintain the arguments which hold a dependency relation with the predicate. From the position of the predicate, the nearest preceding noun argument is selected as the sub-

ject. The following noun arguments are seen as the objects (direct object and indirect objects). A prepositional phrase is represented as a pair of preposition and its argument (e.g., *pp:in:park*). Surface cases of other arguments are represented in their lower case of POS tags. We also distinguish the active and passive voices of a verb. In English for example, we further see whether there exists a verb 'be', which is the head of the chosen predicate. If so, the predicate would be the combination of 'be' and the verb's passive form (e.g., *be_shown*).

Similarly, Chinese predicate-arguement structures are also represented by surface cases. However, Chinese passive voice is little more complex than English. Chinese passive voice is basically marked by character '被 (Bei)'. According to annotation criteria in Chinese Treebank, '被' has two types of POS tag in different situations. The following two examples explain this phenomenon.

- 公司 (company) 被 (Bei) 政府 (government) 列为(list as) 十强(top ten)
- 公司 (company) 被 (Bei) 列为(list as) 十强(top ten)

The POS tag of the characater '被' in the first sentence is 'LB' which is the abbreviation for 'Long Bei'. This stands for the long distance between '被' and the verb '列为'. In contrast, '被' in the second sentence is marked as 'SB' which is the abbreviation for 'Short Bei'. In the case where '被' is directly adjacent to the verb, its POS tag is 'SB'. In other cases, the POS tag of '被' will be 'LB'. As in the first example, '政府', which is a modifier of verb '列为', is labeled as the subject. The argument '公司' which is the modifier of '被/LB' in the first example, and the modifier of the verb '列为' in the second example, becomes the direct object. There is another special case called '把 (Ba)' in Chinese which indicates the direct object:

- 美国 (America) 把 (Ba) 此 (This) 作为(take as) 窗口 (window)

In this example, argument '此' is indicated as the direct object of the verb '作为', even though it appears before the verb. The argument '美国', which is a modifier of '把', became the subject of the verb '作为', even though they have no direct dependency relation.

4.3.4 Using High-quality Predicate-argument Structures

We use the high-quality predicate-argument structures as extra knowledge. During the training and parsing process, besides the basic features for each head-modifier pairs, such as the morphological information, the distance between the head and the modifier. we also extract four types of additional features according to the extra knowledge. The motivation of knowledge usage can be illustrated by the PP-attachment example: "they eat salad with a fork". When a parser fails to learn the strong relation between the verb "eat" and the prepostional phrase "with a fork", due to reasons such as training data size limitation, it may lead to an incorrect analysis where "with a fork" is a modifier of "salad". In this case, extra knowledge can be used as compensations to emphasize the syntactic relation between patterns such as "eat" and "with a fork", and to decrease the relevance between patterns such as "salad" and "with a fork".

Each of these features extracted from the knowledge is described as follows.

Freq the co-occurrence frequency value of the target head-modifier pair in the knowledge.

Pmi the point-wise mutual information (PMI) value for the target head-modifier pair.

PAfreq the frequency of a argument being a certain syntactic role of a predicate, e.g., the frequency of "salad" being an *object* of "eat".

PApmi the PMI value of an argument with its syntactic role and the predicate.

Because using the real value for each feature will lead to the sparseness problem, we use a binned value (i.e., high, middle and low) for all of the feature values calculated from the knowledge.

4.4 Experiments

4.4.1 Experimental Settings

For high-quality predicate-argument structures extraction, 40 million sentences from Chinese Gigaword 5.0 (LDC2011T13) and 400 million sentences from a Chinese Web corpus were used. We only use text written in simplified Chinese. Stanford parser was used to apply dependency parsing on the raw texts from Chinese Gigaword. The training section

of Chinese Treebank 7.0 was used to train the dependency parser and the official development section was used to train a classifier for high-quality dependency selection. For both corpora, we extracted knowledge with three different quality. One was composed with automatic dependencies without applying the dependency selection approach. The other two are composed with 50% and 20% of the high-quality dependencies.

We use the SKP parser [38], which is a graph-based dependency parser as a baseline system. Also we modified the SKP parser to be able to use additional knowledge. Besides the basic features for each edge, we extracted features from the knowledge bases with different quality. We evaluated the parser with additional features from different types of knowledge. Unlabeled attachment score (UAS) was used in evaluation, which calculated the percentage of edges that are correctely judged in dependency trees.

4.4.2 Experimental Results

Table 4.1 shows the experimental results using knowledge constructed from Gigaword and Table 4.2 shows the results using knowledge from the Web. Knowledge (n%) indicates the quality the knowledge. For example, 100% means the knowledge without dependency selection.

We can see from Table 4.1, besides the parser using 20% of high-quality knowledge, the other two settings using additional features from extra knowledge gained an improvement compared to the baseline. Using knowledge without dependency selection has slightly better performance than the parser using 50% of the knowledge.

In Table 4.2, all the parsers using addtional features gained improvement compared to the baseline. Especially the parsers using part of the knowledge (i.e., 50% and 20% of the knowledge). The parser using 50% of the knowledge extracted from the web has the best performance.

4.5 Discussion

As shown in Table 4.1, the parser benifits more when using larger size of knowledge. Even though the dependency selection approach can make knowledge more reliable, it reduces the total size of knowledge and inevitably loses a part of useful information. Texts from Gigaword are generally in the same domain with the training data for the

method	UAS
baseline	86.59%
baseline + giga knowledge (100%)	**87.82**%
baseline + giga knowledge (50%)	86.81%
baseline + giga knowledge (20%)	86.56%

Table 4.1: UAS using knowledge acquired from Chinese Gigaword

method	UAS
baseline	86.59%
baseline + web knowledge (100%)	86.86%
baseline + web knowledge (50%)	**87.81**%
baseline + web knowledge (20%)	87.33%

Table 4.2: UAS using knowledge acquired from Web corpus

dependency parser. When dependency parsing is applied on such texts, relatively good accuracy can be achieved. Thus dependency selection becomes less important when using this kind of texts for knowledge acquisition. On the other hand, Gigaword only contains 40 million sentences which is not a large scale. Losing data size while dependency selection lead to a performance drop.

In the experiments using the Web corpus for knowledge acquisition, dependency selection plays a more crucial role to improve dependency parsing. As shown in Table 4.2, using large-scale knowledge acquired from the Web can improve the parsing performance. Without dependency selection, the knowledge contains many noises. This is because the Web corpus involves all kinds of different domains from the training data for the base parser. As a result, ignoring the noises leads to a slight improvement. The size of Chinese Web corpus is almost ten times larger than Chinese Gigaword, this compensate the information loss during dependency selection. This explains the fact that the biggest improvement is benifited from the high-quality knowledge. On the other hand, excessive selection may cause the decreasing of information diversity. Therefore, knowledge (50%) has a tiny advantage over knowledge (20%).

4.6 Summary

In this chapter, we make use of the automatic aquired predicate-argument structures as an additional knowledge to improve the overall performance of dependency parsing. Automatic parses inevitably contain a large amount of noises, especially for some difficult-to-analyze langauges such as Chinese. For the consideration of the bad effect those noises will bring to the system, the high-quality dependency selection process is also indispensable. Experimental results show that using such additional knowledge can improve the performance of the dependency parser. For the knowledge that potentially contains a large amount of automatic errors, better quality of PAS can offer more benefits to the dependency parser.

Chapter 5

Semantic Role Labeling using High-quality Knowledge

This chapter presents an application of automatic acquired knowledge for semantic role labeling (SRL). Besides basic morphological information, syntactic structures are crucial in SRL. However, it is difficult to learn such information from limited, small-scale, manually annotated training data. Instead of manually increasing the size of annotated data, we use a large amount of automatically extracted knowledge to improve the performance of SRL.

5.1 Introduction

Semantic role labeling (SRL) is regarded as a task that is intermediate between syntactic parsing and semantic analysis in natural language processing (NLP). The main goal of SRL is to extract a proposition from a sentence about *who* does *what* to *whom*, *when, where* and *why*. By using semantic roles, the complex expression of a sentence is then interpreted as an *event* and its *participants* (i.e., predicates and arguments such as *agent, patient, locative, temporal* and *manner*). Unlike syntactic level surface cases (i.e., dependency labels such as subject and object), semantic roles can be regarded as a deep case representation for predicates. Because of its ability to abstract the meaning of a sentence, SRL has been applied to many NLP applications, including information extraction [6], question answering [40] and machine translation [31].

Semantically annotated corpora, such as FrameNet [11] and PropBank [25], make this type of automatic semantic structure analysis feasible by using supervised machine learning methods. Automatic SRL processing has two major drawbacks: firstly, the scale of the training data is quite limited and although manually annotated data such as PropBank is available as training data for learning semantic role prediction models, it is still hard to learn lexical preferences due its limited size. Increasing the size and coverage of this resource for improving the quality of learned models is a time consuming task. Secondly, similar to syntactic analysis such as syntactic dependency parsing, whose performance is highly dependent on preceding analysis such as POS tagging, automatic SRL systems are based on syntactic structures along with lower level information including POS tags and lexical information. As a result, SRL suffers from error propagation from the lower levels of the whole framework. Although some studies use automatic analysis of unlabeled data to enrich the training data to solve the first problem [13], accumulated errors in such automatic analysis inevitably causes negative effects. Especially, for some hard-to-analyze languages such as Chinese, which is difficult to analyze morphologically, the performance of SRL is always limited due to the above two problems.

In this paper, we focus on Chinese SRL and address the problems mentioned above by using high-quality knowledge automatically extracted from a large-scale corpus. Instead of using high level automatic analyses such as semantic roles, we use lower level knowledge because lower level analyses are less erroneous compared to higher level analyses. The additional knowledge can provide not only a rich lexicon but also syntactic information, both of which play crucial roles in SRL. In order to show that automatically extracted knowledge is beneficial, we use predicate-argument structures and case frames (which will be introduced in later sections) in our experiments to validate our claim.

The rest of this chapter is organized as follows. Section 5.2 contains related work. Section 5.3 describes SRL task overflow. Section 5.4 describes the basic features for SRL. Section 5.5 presents a detailed description of our approach Section 5.6 presents the experimental settings along with the results followed by a discussion in Section 5.7. Finally, Section 5.8 contains the summary and future work.

5.2 Related Work

The CoNLL-2009 shared task [15] features a substantial number of studies on SRL that used Propbank as one of the resources. These work can be categorized into two types: joint learning of syntactic parsing and SRL [45, 39], which learns a unique model for syntactic parsing and SRL jointly. This type of framework has the ability to use SRL information in syntactic parsing for improvement, but has a much larger search space during the joint model learning. The other type is called SRL-only task [53, 3], which uses automatic morphological and syntactic information as the input in order to judge which token plays what kind of semantic role. Our work focuses on the second category of SRL. Our framework is based on those used by [3] and [48].

There were also several studies using semi-supervised methods for SRL. One basic idea of semi-supervised SRL is to automatically annotate unlabeled data using a simple classifier trained on original training data [13]. Since there is a substantial amount of error propagation in SRL frameworks, the additional automatic semantic roles are not guaranteed to be of good quality. Contrary to this approach, we only rely on syntactic level knowledge which does not suffer too much from error propagation. Also, some studies assume that sentences that are syntactically and lexically similar are likely to share the same frame-semantic structure [13]. This allows them to project semantic role information to unlabeled sentences using alignments. However, computation of these alignments requires additional information such as word similarity, whose quality is language dependent. Less sparse features capturing lexical information of words can be also used for semi-supervised learning of SRL. Such lexical representation can be learned from unlabeled data [2]. [9] used word similarity learned from unlabeled data as additional features for SRL. Word embeddings have also been used in several NLP tasks including SRL [7]. Instead of using word-level lexical information, our work uses knowledge as syntactic level lexical information. [51] used selectional preferences to improve SRL. This study is similar to our approaches but the quality of selectional preferences was not concerned at all.

In syntactic level of NLP, rich knowledge such as predicate-argument structures and case frames are strong backups for various kinds of tasks. A case frame, which clarifies relations between a predicate and its arguments, can support tasks ranging from funda-

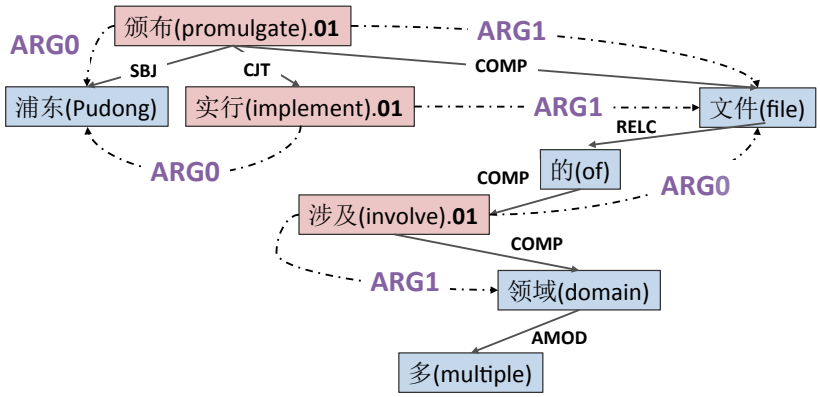

Figure 5.1: SRL task workflow

mental analysis, such as syntactic dependency parsing and word similarity calculation, to multilingual applications, such as machine translation. Japanese case frames have been successfully compiled [21], where each argument is represented as its case marker in Japanese such as 'ga', 'wo', and 'ni'. For the case frames of other languages such as English and Chinese, because there are no such case markers that can help clarify syntactic structures, instead of using case markers like in Japanese, syntactic surface cases (i.e., subject, object, prepositional phrase, etc.) are used for argument representation [19]. Case frames can be automatically acquired using a different method such as Chinese Restaurant Process (CRP) [23] for different languages. In our work, we employ such syntactic level knowledge, which use surface cases as argument representation, to help SRL task. We refer to this as knowledge in this paper.

5.3 SRL Task Description

In previous studies, SRL pipeline[1] can be divided into three main steps: predicate disambiguation (PD), argument identification (AI), and argument classification (AC). Figure 5.1 gives an example of how to apply SRL on raw text.

[1]Predicate identification (PI) was not concerned in this paper because we use the data from CoNLL-2009 shared task, in which the target predicates are given.

5.3.1 Predicate Disambiguation

In the PD step, the main goal is to identify the "sense id" of each given predicate. Because the sense ID for a certain predicate is meaningless for other predicates, classifiers for PD are trained separately for each predicate. We used the part of the feature set proposed by [3] and some additional features. Table 5.1 lists the feature sets used in the PD step. During the prediction, there will be some predicates which have not been seen before in training data. We label the sense of those unseen predicates using the default sense, which is '01' in our work.

5.3.2 Argument Identification

Different from syntactic dependency parsing, given a predicate in a sentence, each token has a possibility to hold a semantic relation with the given predicate. Each token is regarded as an argument candidate. The AI step is mainly to recognize these semantic arguments from the argument candidates.

5.3.3 Argument Classification

In the AC step, which is the last step in the SRL pipeline, each semantic argument is labeled with a semantic role. However, there was some work in which AI and AC step are executed jointly by inducing a new label 'null', which indicates that the token is not a semantic argument of the predicate. As far as we know, there is small amount of debate involving the merging of the AI step and the AC step, especially on whether such merging is beneficial or not. The joint method seems to have an ability to reduce the error propagation from the AI step to the AC step. However, at the same time, since the training samples with label 'null' will consequently outnumber other labels, there is still a drawback during learning. In our work, we apply a separate framework that carries out the AI and AC step in a pipeline since it is much more intuitive. We use features from [3] and [48] along with some new features in AI and AC step. Table 5.2 and Table 5.3 list the features used in each step, in which we use the mark † to indicate the proposed features.

5.4 Basic Features for SRL

In this section, we give a detail description of the features we use in each steps (PD, AI and AC). The features described below are represented as strings or sets of strings. Those features can be categorized into: predicate features, which are extracted from predicate; argument features, which are centered around the potential or identified argument. The rest of the basic features are named as other features.

5.4.1 Predicate Features

The predicate features are basically extracted from the predicate along with some other words that are syntactically related. We take the sentence described in figure 5.2 as an example. The value of a feature is either a string or a set of strings,

PredWord: the surface word of the target predicate, e.g. "实行 (implement)";

PredPOS: the POS tag of the target predicate, e.g. "VV";

PredDeprel: the dependency label of the target predicate, e.g. "CJT";

PredParentWord: the surface word of the parent node of the target predicate, e.g. the parent node of "实行 (implement)" is "颁布 (promulgate)";

PredParentPOS: the POS tag of the parent node of the predicate, e.g. the POS of the parent node of "实行 (implement)" is "VV";

ChildWordSet: the set of surface word of the predicate's children nodes, e.g. the children nodes set of "颁布 (promulgate)" is {"浦东(Pudong)", "实行 (implement)", "文件 (file)"};

ChildPOSSet: the set of POS tags of the children of the target predicate, e.g. the *ChildPOSSet* for the target predicate "颁布 (promulgate)" is {"NN", "VV", "NN"};

ChildDepSet: the set of the dependency labels of the children of the target predicate, e.g. the *ChildDepSet* for the target predicate "颁布 (promulgate)" is {"SBJ", "CJT", "COMP"};

DepSubCat: the concatenation of the dependency labels of predicate 's children with respect to the word order of the original sentence, e.g. the *DepSubCat* for the target predicate "颁布 (promulate)" is "SBJ+CJT+COMP";

Sense: the predicate sense in PropBank, e.g. "实行.01";

5.4.2 Argument Features

In addition to the predicate related features, the AI and AC classifiers also employ features centered around the argument. We also use the sentence in figure 5.2 as an example. For example, we focus on the word "文件" as the target argument.

ArgWord: the surface form of the word of the target argument, e.g. "文件 (file)";

ArgPOS: the POS tag of the target argument, e.g. "NN";

ArgDeprel: the dependency label of the target argument to its head, e.g. "COMP";

LeftSiblingWord: the surface form of the left sibling of the target argument, e.g. "实行 (implement)";

LeftSiblingPOS: the POS tag of the left sibling of the target argument, e.g. "VV";

RightSiblingWord: the surface form of the right sibling of the target argument, e.g. "NULL" (because there is no right sibling node in this case);

RightSiblingPOS: the POS tag of the right sibling of the target argument, e.g. "NULL";

LeftMostDepWord: the surface form of the leftmost dependent of the target argument, e.g. "的 (of)";

LeftMostDepPOS: the POS tag of the leftmost dependent of the target argument, e.g. "DEC"

RightMostDepWord: the surface form of the rightmost dependent of the target argument, e.g. "NULL";

RightMostDepPOS: the POS tag of the rightmost dependent of the target argument, e.g. "NULL";

5.4.3 Other Features

We also extract some features based on the interaction between the predicate and its arguments. These features are categorized as *other features*.

Position: the feature denotes the position of the argument with respect to the predicate. This could be either ‘ Before ’, or ‘ After ’.

DeprelPath: the concatenation of dependency labels and link direction when moving from predicate to argument, e.g. the DeprelPath from “涉及 (involve)” to “实行 (implement)” is “COMP ↑ RELC ↑ COM ↓ CJT”;

IsThePredNearest: a binary feature that indicates whether the predicate is the nearest one to the candidate.

VerbChainHasSubj: a binary feature that is set to true if the predicate verb chain has a subject. The purpose of this feature is to resolve verb coordination ambiguity.

5.5 Proposed Method for SRL

In previous studies, SRL pipeline[2] can be divided into three main steps: predicate disambiguation (PD), argument identification (AI), and argument classification (AC). In the PD step, the main goal is to identify the “sense id” of each given predicate. Because the sense ID for a certain predicate is meaningless for other predicates, classifiers for PD are trained separately for each predicate. We used the part of the feature set proposed by [3] and some additional features. Table 5.1 lists the feature sets used in the PD step. During the prediction, there will be some predicates which have not been seen before in training data. We label the sense of those unseen predicates using the default sense, which is ‘01’ in our work.

[2]Predicate identification (PI) was not concerned in this paper because we use the data from CoNLL-2009 shared task, in which the target predicates are given.

feature	description
PredWord PredPOS PredDeprel PredParentWord PredParentPOS PredParentWord+POS	basic morphologic and syntactic information of the predicate and its parent
ChildWordSet ChildPOSSet ChildDepSet ChildWord+ChildDepSet ChildPOS+ChildDepSet	set feature of the children of predicate
DepSubCat	the concatenation of the dependency labels of predicate's children

Table 5.1: Features for PD

feature	AI	AC	description
PredLemma	•	•	basic morphologic and syntactic information of the predicate
PredPOS	•	•	
PredRel	•	•	
PredLemmaSense	•	•	
Head	•	•	
HeadPOS	•	•	
Pred+HeadWord	•	•	
ArgWord	•	•	basic morphologic and syntactic information of the argument
ArgPOS	•	•	
ArgDeprel	•	•	
DeprelPath	•	•	structural information of the argument in the dependency tree
LeftSiblingWord	•	•	
LeftSiblingPOS	•	•	
RightSiblingWord	•	•	
RightSiblingPOS	•	•	
Position	•	•	
LeftMostDepWord	•	•	
LeftMostDepPOS	•	•	
RightMostDepWord	•	•	
RightMostDepPOS	•	•	
IsThePredNearest	•		binary feature indicating whether the given predicate is the nearest
VerbChainHasSubj	•		binary feature indicating whether there is a dependency label 'SUBJ' between the argument and the predicate

Table 5.2: Basic features for AI and AC

feature	AI	AC	description
†PredContextWord-1/-2/+1+2	•		context information of the predicate
†PredContextPOS-1/-2/+1+2	•		
†PredContextRel-1/-2/+1+2	•		
†ArgContextWord-1/-2/+1+2	•		context information of the argument
†ArgContextPOS-1/-2/+1+2	•		
†ArgContextRel-1/-2/+1+2	•		

Table 5.3: Features for AI and AC part 2 († marks stand for the features we proposed)

Different from syntactic dependency parsing, given a predicate in a sentence, each token has a possibility to hold a semantic relation with the given predicate. Each token is regarded as an argument candidate. The AI step is mainly to recognize these semantic arguments from the argument candidates. In the AC step, which is the last step in the SRL pipeline, each semantic argument is labeled with a semantic role. However, there was some work in which AI and AC step are executed jointly by inducing a new label 'null', which indicates that the token is not a semantic argument of the predicate. As far as we know, there is small amount of debate involving the merging of the AI step and the AC step, especially on whether such merging is beneficial or not. The joint method seems to have an ability to reduce the error propagation from the AI step to the AC step. However, at the same time, since the training samples with label 'null' will consequently outnumber other labels, there is still a drawback during learning. In our work, we apply a separate framework that carries out the AI and AC step in a pipeline since it is much more intuitive. We use features from [3] and [48] along with some new features in AI and AC step. Table 5.2 and Table 5.3 list the features used in each step, in which we use the mark † to indicate the proposed features.

5.5.1 Knowledge Acquisition

We constructed two types of knowledge namely, predicate-argument structures and case frames.

verb	surface case	instance with frequency in original corpus
谢(1)	nsubj	花儿 (flower):14, 花 (flower):22
	ad	都 (all):16, 也 (also):6
谢(2)	nsubj	你们(you):1
	dobj	您(you):8, 我 (me):6
	ad	怎 么(how):8, 多 (very):1
谢(3)	nsubj	大战(battle):1
	dobj	幕 (curtain):6
	ad	圆满(successfully):2, 也 (also):1, 正式 (officially):1
...		

Table 5.4: Examples of Chinese case frames

High-quality Predicate-argument Structure Extraction

Predicate-argument structures (PAS) have been basically acquired from syntactic analyses which varies from phrase chunking to syntactic dependency parsing. For example, English PAS in surface case was acquired in a large scale using a chunking-based system [22]. Some phenomena in Chinese, such as omission and complex grammar, make it intractable to automatically extract PAS only using shallow syntactic analysis, such as chunking. Syntactic dependency parsing is applied for Chinese PAS extraction. Arguments are represented by their syntactic dependency labels (i.e., subject, object, etc.)

Due to various factors, Chinese syntactic dependency parsing is relatively worse in performance compared to that of English, Japanese, etc. However, using an existing treebank, it is possible to train a classifier to acquire high-quality PAS by only using highly reliable syntactic dependencies. As a result, we applied syntactic dependency parsing to large-scale raw corpora and adopted the high-quality syntactic dependency selection approach [19]. Their approach first trains a base parser using a part of the Chinese treebank and then applies syntactic dependency parsing on the raw text of another part of the same treebank. According to the gold-standard annotations, both postive and negative samples are then collected to train a binary classifier, which selects those dependencies more likely to be correct. We also follow their method for the compilation of high-quality PAS, which can provide a massive amount of knowledge.

High-quality Case Frame Compilation

In NLP, at the level of syntax, case frames, compiled from PAS, were proposed as strong backups for various kinds of tasks [21]. For each predicate, all the PAS are clustered into different case frames to reflect different semantic usages. We show an example of case frames for the verb '谢' in Table 5.4, which has multiple meanings. '谢(1)' is the case frame used to represent the sense of 'withering of flower'. Similarly, the sense of '谢' which means 'to thank', the applicable case frame is '谢(2)'. '谢(3)' is the case frame for the sense of 'curtain call'. In other words, case frames are knowledge that solves word sense disambiguation (WSD) by clustering the PAS. We applied the CRP method described by [23] for clustering the high-quality PAS to compile high-quality case frames.

5.5.2 Using Knowledge for SRL

The motivation of using large-scale knowledge is to complement the syntactic information in the limited size of training data. In SRL, an argument may not contain a direct syntactic relation to a given predicate but still plays a semantic role of the predicate. However, this kind of argument can actually form a direct syntactic relation to the predicate when we change the expression of the sentence in other ways. In other words, this kind of argument may hold a direct syntactic relation with the predicate in real world natural languages. This is a frequent phenomenon in multi-verb sentences. Take the sentence in Figure 5.2 as an example.

This sentence can be translated as "promulgated and implemented files involving multiple fields." "文件 (file)" is a child of "颁布 (promulgate)" in the dependency tree and labeled as semantic role "A1" of "颁布 (promulgate)". Even though "文件 (file)" does not have a direct dependency relation with "实行 (implement)", it is still regarded as a semantic role "A1" of "实行 (implement)". Similarly, "文件 (file)" has also a semantic role "A0" of the verb "涉及 (involve)" with no direct dependency relation. However, both direct syntactic dependencies "实行 (implement) 文件 (files)" and "文件 (file) 涉及 (involve)" appear frequently in real world text. Such patterns in surface cases captured from large-scale corpora would be important clues for SRL.

In addition, some special surface cases such as "BA" and "LB/SB" explicitly indicate

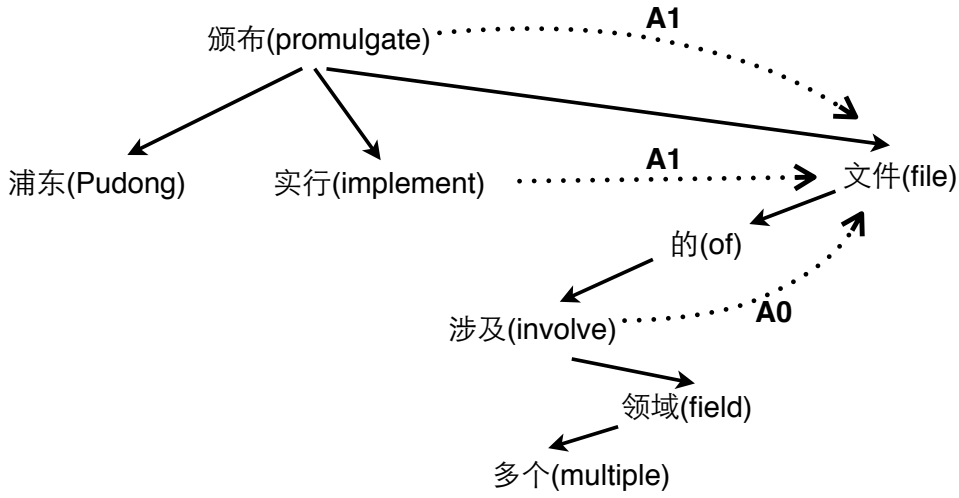

Figure 5.2: Example of dependency and semantic relations (Solid arrows denote syntactic dependencies and dotted arrows denote semantic dependencies)

accusative case and nominative case, which for most of the time is labeled as "A1" and "A0" respectively in PropBank-style SRL specification. "用/以 (use)" is a preposition that strongly indicates the semantic role "MNR" and "在 (at)" is a preposition that always stands for the semantic role "LOC" or "TMP". Therefore, it is promising to use large-scale knowledge as an additional resource.

Predicate-argument Pair Features

From the surface case PAS, we extract four types of additional features, for both AI and AC step. Each of those features is described as follows. We use binned values (i.e., high, middle and low) for all of the feature values calculated from the knowledge.

Freq: the co-occurrence frequency value of a predicate-argument pair without considering the syntactic role of the argument

Pmi: the point-wise mutual information (PMI) value for each predicate-argument pair without considering the syntactic role of the argument

PAfreq: the frequency of a argument being a certain syntactic role of a predicate

PApmi: the PMI value of an argument with its syntactic role and the predicate

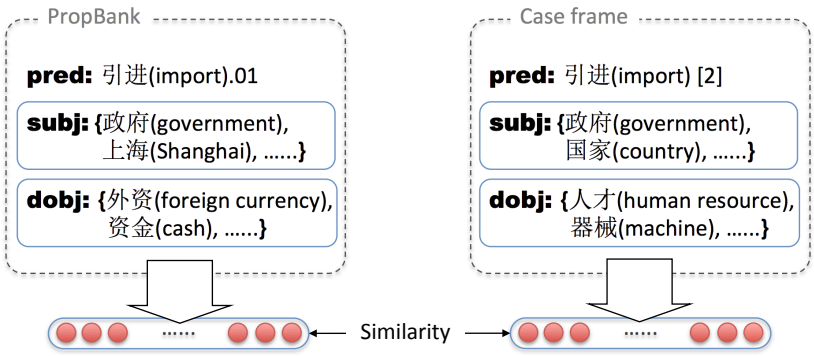

Figure 5.3: Overview of mapping case frames to PropBank sense

Case Frame Features

Case frames are clustered PAS according to each predicate's semantic usage. Therefore, instead of utilizing all the predicate-argument structures, it is also intuitive to use the predicate-argument structures only from the corresponding case frames. So we also create four types of features extracted from case frames.

CFFreq: the **Freq** value calculated only from within the corresponding case frames
CFPmi: the **Pmi** value calculated only from within the corresponding case frames
CFPAfreq: the **PAfreq** value calculated only from within the corresponding case frames
CFPApmi: the **PApmi** value calculated only from within the corresponding case frames

Mapping Case Frames to PropBank Sense

Note that a case frame ID and a PropBank sense ID do not correspond to each other. In practice, the number of case frames is always larger than the number of sense in PropBank for each verb. As a result, a mapping process which aligns case frame id(s) to PropBank verb sense is needed. For example, the verb '引进' may three have PropBank sense ids: '引进.01', '引进.02', '引进.03'. On the other hand, the verb '引进' may have 10 case frames: '引进[1]', '引进[2]', ..., '引进[10]'. Then we calculated the similarity between each PropBank sense and each case frame. A case frame will be finally mapped to the most similar PropBank sense id. Thus the mapping result may look like: '引进.01' correspond to '引进[1]', '引进[3]', '引进[4]'; '引进.02' correspond to '引进[2]', '引进[5]', '引进[10]' etc.

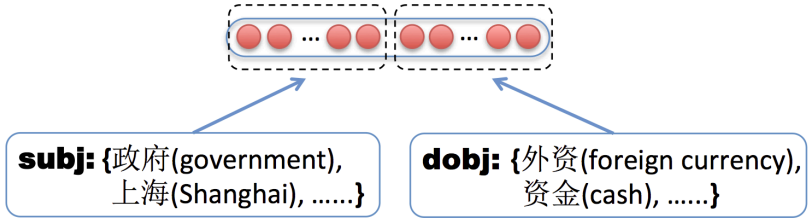

Figure 5.4: Syntactic role vector

We calculate the similarity between a PropBank sense and a case frame by measuring the PAS similarity. As shown in the left part of figure 5.3, for a certain predicate with a sense ID in PropBank, we represent the predicate in each sense by using the collection of all the instances in each syntactic role slot. Each predicate with a sense ID is then transformed into a vector space which we name as PAS vector. The same transformation is applied on case frames. Then cosine similarity between vectors transformed from PropBank sense and case frames is calculated,

As illustrated in figure 5.4, A PAS vector is the concatenation of each syntactic role vector. In our experiments, we only used syntactic role "subj" (subject) and "dobj" (direct object) because there two syntactic roles are considered to be relatively more informative. Each syntactic role vector is calculated by considering all the instances in this syntactic role slot.

$$vec(role) = \frac{\sum_{i=1}^{n} vec(instance_i) \cdot f(instance_i)}{\sum_{i=1}^{n} f(instance_i)} \tag{5.1}$$

As shown in the formula 5.1, a syntactic role vector $vec(role)$ is calculated by: first, taking the summation of frequency-weighted instance vectors; seconde, averaged by the summation of instance frequency. An instance vector is represented by its word embedding, which can be measured by any off-the-shelf toolkit such as Word2vec which we employed in the experiments. The raw text used for word embedding training

5.6 Experiments

5.6.1 Experimental Settings

For large-scale knowledge acquisition, 40 million sentences from Chinese Gigaword 5.0 (LDC2011T13)[3] were used. 400 million sentences from Chinese Web corpus were also used in our experiments.

For the high-quality dependency selection approach in the knowledge construction pipeline, the Stanford parser was used to apply syntactic dependency parsing on the raw texts from Chinese Gigaword. The training section of Chinese Treebank 7.0 was used to train the dependency parser and the official development section was used to train a classifier for high-quality dependency selection. Judging whether the automatic dependencies are reliable can be regarded as a binary classification problem, for which we utilized support vector machines (SVMs). Specifically, we employed SVM-Light[4] with a linear kernel to select high-quality dependencies from large-scale automatic dependency parses on the Chinese Gigaword for knowledge construction. Using official evalution section of CTB 7.0, we evaluated the quality of thoses selected dependencies using unlabeled attachment score (UAS), which calculates the percentage of correctly indentified dependency heads.

For SRL, we used the Chinese section of CoNLL-2009 shared task data for experiments. Automatically obtained morphological and syntactic information (the columns begin with "P") was used. PD and AI, AC step are regarded as multi-class classification problems. We employed OPAL[5] to solve this problem. We set the options as follows: polynomial kernel with degree 2; passive aggressive I learner; 20 iterations. The SRL system without using additional knowledge was used as a baseline. To examine the effect of different quality of knowledge, we used different set of PAS which was extracted under different dependency selection thresholds (20%, 50%, w/o selection). The official script provided on the CoNLL-2009 shared task website was used for evaluation.

In practice, we apply the Stanford parser for syntactic analysis and compiling case frames. For the consideration of SRL on large-scale raw text, we also tested the effec-

[3] We only used sentences written in simplified characters in Chinese Gigaword.

[4] http://svmlight.joachims.org/

[5] http://www.tkl.iis.u-tokyo.ac.jp/~ynaga/opal/

tiveness of SRL system using Stanford dependencies We prepared another version of CoNLL-2009 shared task data by substituting the syntactic dependencies and syntactic roles using the Stanford parser. Other settings were identical with the experiment on the original CoNLL-2009 shared task data.

5.6.2 Experimental Results

	w/o selection	**select 50%**	**select 20%**
UAS	0.677	0.824	0.920

Table 5.5: Precision of selected dependencies under different criteria

Tabel 5.5 shows the quality of selected dependencies using different selection criteria. The precision of automatic syntactic dependencies increases when we lower the recall. The precision over 90% can be achieved when the recall is down to around 20%.

Table 5.6 shows our experimental results using CoNLL-2009 shared task original data. Knowledge (n%) indicates that the top n% (according to the classifier) of the automatically extracted knowledge was used. '100%' means that dependency selection step was not applied

Our baseline system outperforms as well as the best system (which has the F-value of 78.6) in CoNLL-2009 shared task. As we can see from the result, experimental settings using Gigaword knowledge extracted from automatic parses without any selection (100%) has worse performance than the baseline. Using 50% of the Gigaword knowledge does not outperform the baseline either. However, using 20% of the knowledge has a slight improvement in Chinese SRL task. In the experimental settings using knowledge from the web, using all kinds of knowledge has gained benifits. However, selecting high-quality knowledge also has a positive effect on overall SRL performance.

Table 5.7 shows the experimental results using the data set with Stanford dependencies Compared to the original CoNLL-2009 shared task data which uses different style of syntactic roles, using this data set benefits more from using the CoNLL-2009 shared task original data. Selecting high-quality knowledge brings significant overall performance improvement both when using Gigaword knowledge and Web knowledge.

method	precision	recall	F1
baseline	82.66%	76.47%	79.44
baseline + giga knowledge (100%)	82.10%	76.38%	79.14
baseline + giga knowledge (50%)	82.44%	76.44%	79.32
baseline + giga knowledge (20%)	82.65%	76.53%	**79.47**
baseline + www knowledge (100%)	82.64%	76.59%	79.50
baseline + www knowledge (50%)	82.73%	76.52%	79.51
baseline + www knowledge (20%)	82.85%	76.60%	**79.60**

Table 5.6: Evaluation results of Chinese SRL using CoNLL2009 shared task data.

method	precision	recall	F1
baseline	80.66%	72.98%	76.63
baseline + giga knowledge (100%)	79.86%	72.72%	76.12
baseline + giga knowledge (50%)	80.40%	73.04%	76.54
baseline + giga knowledge (20%)	80.73%	73.32%	****76.85**
baseline + www knowledge (100%)	80.71%	73.17%	76.76
baseline + www knowledge (50%)	80.81%	73.27%	*76.86
baseline + www knowledge (20%)	80.94%	73.20%	***76.88**

Table 5.7: Evaluation results of Chinese SRL using Stanford dependencies. The ** mark and * mark mean that the result is regarded as significant (with a p value < 0.01 and a p value < 0.05 respectively) using McNemar's test.

5.7 Discussion

In the experiments using original CoNLL-2009 shared task data, the experiment settings using Gigaword knowledge extracted from automatic parses without any selection (100%) has worse performance than the baseline. This is mainly due to the fact that SRL system is sensitive to those automatic analyzing errors which may bring negative effect. Using 50% of the Gigaword knowledge performs slightly better than without selection, but still not as good as the baseline system. Using 20% of the knowledge has a slight improvement than the baseline but due to the size decreasing fact, the improvement is not so obvious. In the experimental settings using knowledge from the web, using all kinds of knowledge has gained benifits. Selecting high-quality knowledge also has a positive effect on overall SRL performance.

In the original CoNLL-2009 shared task data, the dependency head column and the syntactic label column are annotated using MaltParser[6], which has different dependency style and uses a totally different set of syntactic roles compared to the Stanford parser. Since the knowledge bases are built from the output of the Stanford parser, lack of consistency in syntactic information leads to less effectiveness when using extra knowledge. That explains why the results are insignificant even when using high-quality knowledge.

In the experiments using the data set with Stanford dependencies. For the system using Gigaword knowledge, the experimental settings without dependency seleciton and selecting 50% of auto-parses does not outperform the baseline system. Using more strict threshold to select only 20% of the auto-parses gains a significant improvement compared to the baseline system. When using the Web corpus, which is larger in scale, all the knowledge in different quality shows improvement compared to baseline system. Also, using higher quality knowledge gains more benifits during SRL. The inevitable noises from the Web make it slightly less significant than the Gigaword knowledge. Compared to the original CoNLL-2009 shared task data which uses different style of syntactic roles, using this version of data set benefits more from extra knowledge because of the syntactic role consistency.

[6] `http://www.maltparser.org/`

5.8 Summary

In this chapter, we have used high-quality knowledge to improve Chinese SRL. The result showed that this kind of knowledge has a positive effect on the SRL performance. The quality of knowledge turns out to be an important factor in such a semi-supervised learning approach.

In the future, we plan to make use of other low level knowledge such as word embeddings [7] or word clusters [26], which can be complementary to our syntactic level knowledge. Since recent SRL approaches are mostly point-wise, i.e., features are extracted from pairs of the predicate and an argument candidate. We plan to design a higher order system to capture more global features. Also, reranking is widely utilized in many SRL systems and we plan to combine our surface case knowledge with a reranker, in order to further improve Chinese SRL. Finally, we plan to experiment on different languages and compare the effectiveness of knowledge for different languages.

Chapter 6

High-quality Semantic Role Selection for Deep Case Frame Construction

In this chapter, we present a framework for the construction of a new variety of case frames which are based on automatic semantic roles. This framework extracts high-quality semantic roles in PropBank style for each predicate. Along with the predicate, acquired semantic roles can be formed into predicate-argument structures (PASs) which are represented in deep case compared to the PASs composed of syntactic roles. Also, using automatic predicate-disambiguation process, PASs can be automatically assigned to different frames. This type of case frames can solve the role disambiguation problems for languages that have flexible word order and lack explicit case markers.

6.1 Introduction

As introduced in Chapter 3, In order to improve the fundamental processes for text understanding, case frames play a very essential role as a backup. Case frames are composed by using automatic dependency parses especially those indicate a predicate-argument relation. That is to say, we center the predicate and acquire words with a direct dependency relation with the target predicate as arguments. Considering the negative effect brought by the erroneous parses produced by a dependency parser, automatic high-quality de-

pendency selection process is applied. Also, to make a distinction between the different usages of a target predicate, unsupervised clustering is applied to assign PAS into different semantic frames, which is the final specification of case frames.

Such knowledge (e.g., PAS and case frames) is successfully acquired by an automatic framework, and has proven to have a positive impact when supporting other tasks such as dependency parsing and semantic role labeling. However, there are major issues for languages that have flexible word order but lack explicit case markers. One representative example is Chinese. For languages like Chinese, even though we can use the syntactic roles (e.g., subject, direct object) for case representation, it is still ambiguous to represent the PAS at a semantic level. We will describe this kind of problem in detail in the following section.

As a result, instead of using syntactic roles for PAS extraction, we utilize semantic roles for better representation. Therefore, the Semantic Role Labeling (SRL) task is induced to the whole framework of case frame compilation. The overall performance of SRL is limited by many aspects such as error propagation. Similar to dependency selection approach, it is also possible to select high-quality semantic roles for each predicate. Those selected semantic roles can also be used for knowledge construction.

The rest of the chapter is split into the following sections: Section 6.2 describes the major issues when using syntactic roles for knowledge acquisition. Section 6.3 describes the approach of high-quality semantic role selection. Section 6.4 details the deep case frame construction framework and the usage for SRL improvement. Section 6.1 describes the method that makes use of deep case frame for SRL improvement. Section 6.6 shows the experimental settings along with the experimental results. Finally, we summrize this chapter in Section 6.7.

6.2 Main Problem

In previous works [21], case frames for Japanese are composed of all the instances and its corresponding case marker. For example, all the instances in “が” case are basically the “subject” of the given predicate. instances in “を” case are basically the “direct object” of the given predicate. Other cases like “に” can indicate the “location”, “time” or “direction” etc. During the automatic PAS extraction for Japanese, there are also

ambiguous case makers that can represent multiple cases. The most common one, for example, is the "は" case in Japanese. This case marker always functions as a topic marker. The argument in "は" case is normally emphasized as the topic of the sentence. It can be equal to either "が" case or "を" case, and sometimes "に" case. To avoid such ambiguous cases, one can simply discard all the instances in "は" case to make the case frame more precise.

For languages that lack such case markers (English and Chinese etc.), case frames are composed of automatic syntactic roles [19]. Such syntactic roles include "subject", "direct object", "indirect object" and "prepositional phrases" etc. Such surface cases have limitations on case representation especially for Chinese. Take the following sentences as examples.

1 苹果 (apples) 我 (I) 吃了 (eaten) 很多 (a lot).
2 我 (I) 苹果 (apples) 吃了 (eaten) 很多 (a lot).
3 我 (I) 吃了 (eaten) 很多 (a lot) 苹果 (apples).

These three sentences have the same meaning: "I have eaten a lot of apples." However, as we can see from the sentences, the word "苹果 (apple)" which is a direct object of "吃了 (eaten)", and the word "我 (I)" can be filled in various word orders. This order-free phenomenon is quite different from English because the word order in English is relatively fixed. Also, because omission occurs frequently in Chinese, the following sentences are also commonly used.

4 我 (I) 吃了 (eaten) 很多 (a lot).
5 苹果 (apples) 吃了 (eaten) 很多 (a lot).

where the first sentence means "I have eaten a lot of (something)." while the second means "(I) have eaten a lot of apples." Without considering the actual meaning of "我 (I)" and "苹果 (apples)", both of these words are labeled as "subject" in surface case following the syntactic grammar. Looking at these two sentences, if one tries to figure out which "subject" is actually in *Nominative Case* (which stands for the person/thing who provide the action) and which "subject" is in *Accusative Case* (which stands for the thing/person who receive/suffer from the action), it is always problematic because of the flexible word order and omission.

Similar phenomena also occur in Japanese and make it difficult to analyze as well. However, in case of Japanese, it is possible to make use of the morphemes attached to the predicate. For example, if one tries to express the same meaning in sentence [4] and [5], it will look like:

6 私が (I)　たくさん (a lot)　食べた (eaten).

7 りんごが (apples)　たくさん (a lot)　食べられた (eaten).

there is always an additional morpheme (e.g., "られた") attached to the predicate in order to indicate its voice. In the above example, sentence [6] can be regarded as active voice and sentence [7] is in passive voice. Also, these types of morphemes can help a parser judge the transitivity of the predicate for further case analysis.

Unfortunately, Chinese is a language that lacks morpheme information. There are very few such markers that indicate the transitivity, voice or tense etc. This makes it almost impossible for a system to automatically recognize the ambiguous syntactic roles. To solve this problem, based on the syntactic analysis, we apply a Semantic Role Labeling (SRL) process to discover a deeper level case representation.

6.3 High-quality Semantic Role Selection

6.3.1 Overview of Semantic Role Selection

Compared to syntactic analysis, SRL is mainly used to clarify deeper-level semantic relations (e.g., [*who*] do [*what kind of*] thing to [*whom*] in [*what time*]) in the sentence, which has better representation for predicate-argument relations. On the other hand, this task is always based on the approaches in previous levels such as morphological analysis and syntactic parsing. Especially, the information provided by syntactic parsing is crucial to achieve a good performance in SRL. A SRL system also suffers from the training data size issue as most of the machine learning approaches do. Extensive human effort is required in order to construct such training data. Sometimes, the requirement for annotators can be higher than those for syntactic analysis. These factors along with the automatic analyzing errors propagated from the lower-level analyses make it almost impossible for a SRL system to achieve a high performance.

Similar to the previous work described in Jin et al. (2013), instead of using all the SRL outputs, we propose to use only those auto-parsed semantic roles with a high reliability.

6.3.2 Proposed Method for Semantic Role Selection

In particular, the standard training section of the human-annotated data is used to train a base SRL model (which include three sub-models for predicate sense disambiguation (PD), argument identification (AI) and argument classification (AC)), and then another part of the human-annotated data is used to apply SRL analysis using the base model. From the automatic parses, we acquired training data for semantic role selection by collecting each semantic role. We then judge correctness of each semantic role according to the gold standard annotations. All correct dependencies were used as positive training examples for dependency selection and vice versa.

Judging whether a semantic role is reliable can be regarded as a binary classification problem. We defined sets of features and use SVM to solve this problem. We list the features for semantic role selection as follows:

SRL features: We use the feature set used in SRL systems [20]. It contains predicate features that are extracted from the target predicate; argument features which are extracted from each candidate argument;

Predicted labels: Besides the features for SRL base system, we also consider the automatic SRL output as another type of important clue for semantic role selection. A system may tend to judge certain types of semantic roles correctly with high precision, but might tend to judge others with low precision. Thus we use the automatic semantic role labels produced by the base SRL system as an additional features.

Knowledge features: We also use the additional knowledge which has positive effect on SRL task.

6.4 Deep Case Frame Construction

In previous work [19] case frames are constructed by the PASs that are composed of syntactic roles. Syntactic roles are used to represent the surface case for each argument. Due

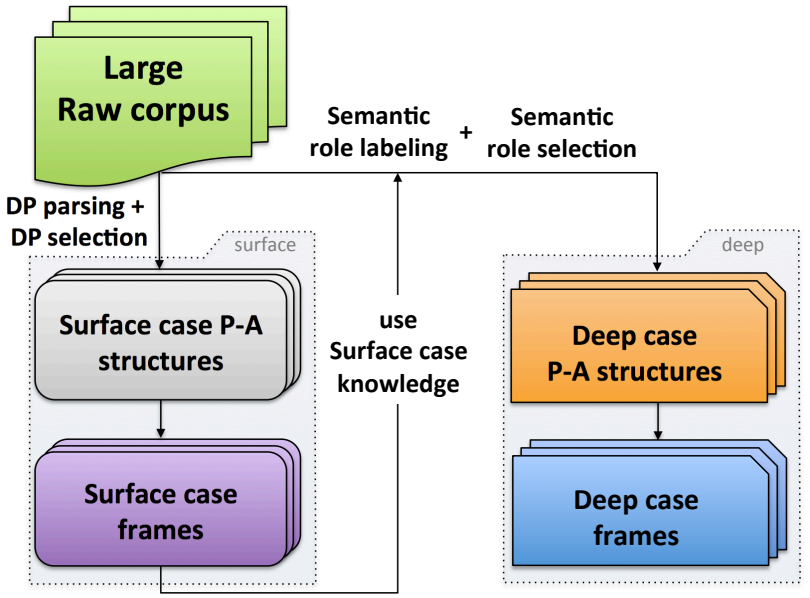

Figure 6.1: Deep case frame construction overview

to the major issues described in section 6.2, case frames constructed using surface cases may also be problematic. For example, for the predicate "吃 (eat)", both the argument "苹果 (apple)" and "我 (I)" are assigned into the same surface case "subject". If one tries to use this kind of surface case knowledge for tasks that require semantic information, such as machine translation (MT), it may cause serious problems and lead to a performance drop. So we propose to construct deep case frames that are relatively more representative than the surface case. By deep case, we mean using the semantic roles for case frame construction.

We illustrate the concept of deep case frame construction in Figure 6.1. In the first stage, we constructed surface case frame using high-quality syntactic information from a raw corpus. Using the surface case frames, we gain an improvement for the SRL system that is one of the most essential processes for deep case frame construction. After applying SRL on the same raw corpus, we apply a semantic role selection described in section 6.3.

It is necessary to mention that there is a predicate identification (PI) step before predicate disambiguation (PD). In PI step, the system needs to identify which words are predicates in a sentence. We did not apply this process in Chapter 5 because the predicates

are given in the training data described in Chapter 5. However, a high performance for PI can be achieved by using simple rules. For example, we regard every word with a POS tag begin with "V" as predicate.

Using PD in the SRL system, a predicate is automatically assigned into different frames. This is equivalent to the unsupervised clustering for surface case frames and thus no additional clustering processes are required. After argument identification and argument classification, we only use those semantic roles with high reliability. For each predicate with different frame IDs, we collect all the high-quality semantic roles to compose the deep case frames.

6.5 Improve SRL using Deep Case Frames

Syntactic information such as dependencies is essential for an SRL system. In Chapter 5, we used surface case frames to provide additional information especially syntactic-level information to an SRL system and gained slight improvement. Deep case frames are compiled using automatic semantic roles which use semantic-level representation. Thus, we consider that using deep case frames as additional knowledge has more direct impact on the performance of SRL. Similar to the method in Chapter 5, we also propose a set of features extracted from deep case frames.

The first four features do not concern the predicate sense. These features are similar to the predicate-argument pair features described in Chapter 5.

SRFreq: the co-occurrence frequency value of a predicate-argument pair without considering the semantic role of the argument

SRPmi: the point-wise mutual information (PMI) value for each predicate-argument pair without considering the semantic role of the argument

SRPAfreq: the frequency of a argument being a certain semantic role of a predicate

SRPApmi: the PMI value of an argument with its semantic role and the predicate

The rest four features are similar to the case frame features described in Chapter 5. However, because the deep case frame ID is identical to the PropBank ID, no mapping processes are needed.

DCFFreq: the **SRFreq** value calculated only from within the corresponding deep case frame

DCFPmi: the **SRPmi** value calculated only from within the corresponding deep case frame

DCFPAfreq: the **SRPAfreq** value calculated only from within the corresponding deep case frame

DCFPApmi: the **SRPApmi** value calculated only from within the corresponding deep case frame

6.6 Experiments

6.6.1 Experimental Settings

For semantic role selection, the training section of CoNLL2009 shared task data is used to train a basic SRL model. The development section in CoNLL2009 shared task-data is used to apply automatic SRL and obtain training data for the semantic role selector. We also prepared another data set by substituting the dependencies and syntactic roles in the original data with Stanford auto parses. We evaluate the semantic role selection model using precision and recall:

$$precision = \frac{\#\,of\ correctly\,retrieved\,semanticroles}{\#\,of\ semanticroles\,retrieved}$$

$$recall = \frac{\#\,of\ correctly\,retrieved\,semanticroles}{\#\,of\ correct\,semanticroles\,in\,gold\,standard\,data}$$

For deep case frame construction, we used Stanford parser for syntactic analysis. The SRL system from Jin et al. (2015) is used to apply deeper case analysis. We applied the framework on 40 million sentences from Chinese Gigaword 5.0 and 400 million sentences from Chinese Web Corpus.

For the SRL system, we used the CoNLL2009 shared task data using Stanford dependencies. We constructed deep case frames of different quality (20%, 50%, w/o selection) to extract extra features to support the base SRL system.

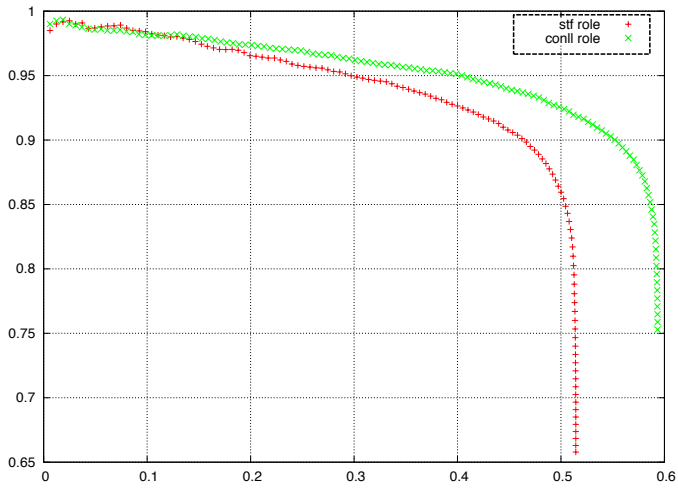

Figure 6.2: Precision-recall curve of semantic role selection

method	precision	recall	F1
baseline	80.66%	72.98%	76.63
baseline + surface case frames (100%)	79.86%	72.72%	76.12
baseline + surface case frames (50%)	80.40%	73.04%	76.54
baseline + surface case frames (20%)	80.73%	73.32%	76.85
baseline + deep case frames (100%)	81.22%	73.55%	77.19
baseline + deep case frames (50%)	81.30%	73.70%	77.31
baseline + deep case frames (20%)	81.57%	73.68%	**77.42**

Table 6.1: Evaluation results of Chinese SRL using different knowledge

6.6.2 Experimental Results

Figure 6.2 shows the precision-recall curves of the selected semantic roles using SVMs. The green curve indicates the results obtained using the original CoNLL-2009 shared task data. The red curve indicates the results using the CoNLL-2009 shared task data by replacing the existing syntactic roles with Stanford dependencies. Note that the base SRL system has a lower precision than described in chapter 5. This is because we did not count predicate sense for evaluation. As we can see from the results, we can successfully select high-quality semantic roles among automatic SRL outputs. For both settings, when

we choose 20% of the semantic roles, we can achieve a precision above 90%.

Table 6.1 shows the performance of SRL systems using different knowledge. As we can see from the results, using deep case frames gained more improvement than using surface case frames. This is because deep case frames are able to directly provide semantic-level information that is insufficient in the training data of the baseline SRL system. Furthermore, the results show that high-quality semantic role selection approach has a positive effect on SRL improvement.

6.7 Summary

In this chapter, we proposed a semantic role selection approach for deep case frame construction. To address the case ambiguity problem in surface case, especially for Chinese, we utilized automatic semantic roles produced by an SRL system for better representation. The experimental results show a promising result for high-quality semantic role selection. Also, using high-quality deep case frames that are composed of semantic roles can significantly improve the baseline SRL system.

Chapter 7

Conclusion

7.1 Summary

In this thesis, we proposed a language independent framework to acquire high-quality knowledge that can further support different type of NLP tasks.

In Chapter 2, we proposed a classification approach to select high-quality dependencies from automatic parses. We created a set of features that consider context and tree information in for selecting dependencies from parsed sentences. This approach can be applied on different languages despite of the different lingual characteristics. The experiments showed that our method works for in-domain parses as well as out-of-domain parses.

In Chapter 3, we propose a framework for automatically constructing case frames for multiple languages. High-reliable predicate-argument structures from raw corpora and well clustered instances with similar semantic features to produce the final case frames.

In Chapter 4, we make use of high quality automatic acquired PAS as additional knowledge to improve the dependency parsing which is one of the most important fundamental tasks. Automatic parses inevitably contain large amount of noises, especially for some difficult-to-analyze languages such as Chinese. In order to avoid the bad effect brought by the noise to a dependency parser, the high-quality dependency selection process is also indispensable. Experimental results show that using such high-quality additional knowledge can improve the performance of the dependency parser. For the knowledge that tends to contain large amount of automatic errors (such as Web text),

higher-quality PASs improve the base parser more.

In Chapter 5, we used high-quality knowledge to improve Chinese SRL that is an intermediate between syntactic parsing and deep level semantic analysis. The knowledge we used in the experiment includes high-quality PASs and case frames which are a clustered version of PASs. The result showed that this kind of knowledge has a positive effect on the SRL performance. The quality of such knowledge is also considered to be an important factor in such a semi-supervised learning approach for SRL.

In Chapter 6, to address the case ambiguous problem in surface case, especially for Chinese, we proposed a semantic role selection approach to construct case frames represented in deep case. We utilized automatic semantic roles produced by an SRL system and selected those semantic roles with high quality. High-quality semantic roles were then utilized to compile deep case frames which were further used to improve SRL. The experimental results show a promising result for high-quality semantic role selection in practice. Also, using high-quality deep case frames as additional knowledge can significantly improve the SRL system.

7.2 Future Work

Since we can use this framework to construct knowledge in different languages, it is also possible to discover the usability in cross-lingual situations. We plan to investigate the possibility for the knowledge in multiple languages to improve machine translation (MT).

We also plan to make use of other low level knowledge such as word embeddings [7] or word clusters [26] to improve the fundamental analyses such dependency parsing and SRL. The recent SRL approaches are mostly point-wise. Features are extracted from only pairs of the predicate and an argument candidate. We plan to design a higher order system to capture more global features following the idea of higher-order dependency parsing. Also, reranking is widely utilized in many SRL systems and we plan to combine our surface case knowledge with a reranker in order to further improve Chinese SRL.

Since this framework uses fundamental analyses to acquire knowledge, and uses acquired knowledge to improve these fundamental analyses as well, we plan to use a bootstrapping strategy to gradually improve all the sub-tasks involved in this framework.

Bibliography

[1] M. Bansal and D. Klein. Web-scale features for full-scale parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 693–702, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

[2] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin. A neural probabilistic language model. 3:1137–1155, February 2003.

[3] A. Björkelund, L. Hafdell, and P. Nugues. Multilingual semantic role labeling. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL 2009): Shared Task*, pages 43–48, Boulder, Colorado, June 2009. Association for Computational Linguistics.

[4] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the 11th Anuual Conference on Computational Learning Theory*, pages 92–100.

[5] W. Chen, J. Kazama, K. Uchimoto, and K. Torisawa. Improving dependency parsing with subtrees from auto-parsed data. In *Proceedings of EMNLP 2009*, pages 570–579, 2009.

[6] J. Christensen, Mausam, S. Soderland, and O. Etzioni. Semantic role labeling for open information extraction. In *Proceedings of the NAACL HLT 2010 First International Workshop on Formalisms and Methodology for Learning by Reading*, pages 52–60, Los Angeles, California, June 2010. Association for Computational Linguistics.

[7] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. Natural language processing (almost) from scratch. 12:2493–2537, August 2011.

[8] F. Dell'Orletta, G. Venturi, and S. Montemagni. Ulisse: an unsupervised algorithm for detecting reliable dependency parses. In *Proceeding of CoNLL 2011*, pages 115–124, 2011.

[9] K. Deschacht and M.-F. Moens. Semi-supervised semantic role labeling using the latent words language model. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 21–29, Singapore, August 2009. Association for Computational Linguistics.

[10] J. Eisner. Bilexical grammars and their cubic-time parsing algorithms. In H. Bunt and A. Nijholt, editors, *Advances in Probabilistic and Other Parsing Technologies*, pages 29–62. Kluwer Academic Publishers, Oct. 2000.

[11] C. J. Fillmore, C. Wooters, and C. F. Baker. Building a large lexical databank which provides deep semantics. In B. Tsou and O. Kwong, editors, *Proceedings of the 15th Pacific Asia Conference on Language, Information and Computation*, Hong Kong, 2001.

[12] D. Flannery, Y. Miayo, G. Neubig, and S. Mori. Training dependency parsers from partially annotated corpora. In *Proceedings of IJCNLP 2011*, pages 776–784, 2011.

[13] H. Fürstenau and M. Lapata. Semi-supervised semantic role labeling. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, pages 220–228, Athens, Greece, March 2009. Association for Computational Linguistics.

[14] Y. Goldberg and M. Elhadad. Precision-biased parsing and high-quality parse selection. 2012. `http://www.cs.bgu.ac.il/~yoavg/publications/emnlp2012confidencea.pdf`.

[15] J. Hajič, M. Ciaramita, R. Johansson, D. Kawahara, M. A. Martí, L. Màrquez, A. Meyers, J. Nivre, S. Padó, J. Štěpánek, P. Straňák, M. Surdeanu, N. Xue, and Y. Zhang. The conll-2009 shared task: Syntactic and semantic dependencies in

multiple languages. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL 2009): Shared Task*, pages 1–18, Boulder, Colorado, June 2009. Association for Computational Linguistics.

[16] C. Hashimoto, K. Torisawa, S. D. Saeger, J. Kazama, and S. Kurohashi. Extracting paraphrases from definition sentences on the web. In *Proceedings of ACL 2011*, pages 1087–1097, 2011.

[17] M. Iwatate. *Development of Pairwise Comparison-based Japanese Dependency Parsers and Application to Corpus Annotation*. PhD thesis, NAIST, Japan, 2012.

[18] G. Jin, D. Kawahara, and S. Kurohashi. High quality dependency selection from automatic parses. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 947–951, 2013.

[19] G. Jin, D. Kawahara, and S. Kurohashi. A framework for compiling high quality knowledge resources from raw corpora. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 109–114, 2014.

[20] G. Jin, D. Kawahara, and S. Kurohashi. Chinese semantic role labeling using high-quality syntactic knowledge. In *Proceedings of the Eighth SIGHAN Workshop on Chinese Language Processing*, pages 120–127, Beijing, China, July 2015. Association for Computational Linguistics.

[21] D. Kawahara and S. Kurohashi. A fully-lexicalized probabilistic model for Japanese syntactic and case structure analysis. In *Proceedings of HLT-NAACL 2006*, pages 176–183, 2006.

[22] D. Kawahara and S. Kurohashi. Acquiring reliable predicate-argument structures from raw corpora for case frame compilation. In *Proceedings of LREC 2010*, pages 1389–1393, 2010.

[23] D. Kawahara, D. Peterson, O. Popescu, and M. Palmer. Inducing example-based semantic frames from a massive amount of verb uses. In *Proceedings of the 14th*

Conference of the European Chapter of the Association for Computational Linguistics, pages 58–67, Gothenburg, Sweden, April 2014. Association for Computational Linguistics.

[24] D. Kawahara and K. Uchimoto. Learning reliability of parses for domain adaptation of dependency parsing. In *Proceedings of IJCNLP 2008*, pages 709–714, 2008.

[25] P. Kingsbury and M. Palmer. From treebank to propbank. In *Language Resources and Evaluation*, 2002.

[26] T. Koo, X. Carreras, and M. Collins. Simple semi-supervised dependency parsing. In *Proceedings of ACL-08: HLT*, pages 595–603, Columbus, Ohio, June 2008. Association for Computational Linguistics.

[27] T. Koo and M. Collins. Efficient third-order dependency parsers. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1–11, Uppsala, Sweden, July 2010. Association for Computational Linguistics.

[28] A. Korhonen, Y. Krymolowski, and T. Briscoe. A large subcategorization lexicon for natural language processing applications. In *Proceedings of the 5th international conference on Language Resources and Evaluation*, pages 345–352, 2006.

[29] C. Kruengkrai, K. Uchimoto, J. Kazama, Y. Wang, K. Torisawa, and H. Isahara. An error-driven word-character hybrid model for joint chinese word segmentation and pos tagging. In *Proceedings of ACL-IJCNLP 2009*, pages 513–521, 2009.

[30] D. Lin. Automatic retrieval and clustering of similar words. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Volume 2*, pages 768–774, 1998.

[31] D. Liu and D. Gildea. Semantic role features for machine translation. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 716–724, Beijing, China, August 2010. Coling 2010 Organizing Committee.

[32] X. Ma and F. Xia. Dependency parser adaptation with subtrees from auto-parsed target domain data. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 585–590, Sofia, Bulgaria, August 2013. Association for Computational Linguistics.

[33] D. McClosky, E. Charniak, and M. Johnson. Effective self-training for parsing. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 152–159, New York City, USA, June 2006. Association for Computational Linguistics.

[34] R. McDonald and J. Nivre. Characterizing the errors of data-driven dependency parsing models. In *Proceedings of EMNLP-CoNLL 2007*, pages 122–131, 2007.

[35] R. McDonald and F. Pereira. Online large-margin training of dependency parsers. In *Proceedings of EACL 2006*, pages 81–88, 2006.

[36] R. McDonald, F. Pereira, K. Ribarov, and J. Hajič. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, pages 523–530, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics.

[37] A. Mejer and K. Crammer. Are you sure? confidence in prediction of dependency tree edges. In *Proceedings of HLT-NAACL 2012*, pages 573–576, 2012.

[38] D. K. Mo Shen and S. Kurohashi. A reranking approach for dependency parsing with variable-sized subtree features. In *Proceedings of 26th Pacific Asia Conference on Language Information and Computing*, pages 308–317, 2012.11.8.

[39] R. Morante, V. Van Asch, and A. van den Bosch. Joint memory-based learning of syntactic and semantic dependencies in multiple languages. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL 2009): Shared Task*, pages 25–30, Boulder, Colorado, June 2009. Association for Computational Linguistics.

[40] L. A. Pizzato and D. Mollá. Indexing on semantic roles for question answering. In *Coling 2008: Proceedings of the 2nd workshop on Information Retrieval for*

Question Answering, pages 74–81, Manchester, UK, August 2008. Coling 2008 Organizing Committee.

[41] R. Reichart and A. Rappoport. An ensemble method for selection of high quality parses. In *Proceedings of ACL 2007*, pages 408–415, 2007.

[42] R. Reichart and A. Rappoport. Automatic selection of high quality parses created by a fully unsupervised parser. In *Proceedings of CoNLL 2009*, pages 156–164, 2009.

[43] S. D. Saeger, K. Torisawa, M. Tsuchida, J. Kazama, C. Hashimoto, I. Yamada, J. H. Oh, I. Varga, and Y. Yan. Relation acquisition using word classes and partial patterns. In *Proceedings of EMNLP 2011*, pages 825–835, 2011.

[44] K. Sagae and J. Tsujii. Dependency parsing and domain adaptation with lr models and parser ensemble. In *Proceedings of EMNLP-CoNLL 2007*, pages 408–415, 2007.

[45] B. Tang, L. Li, X. Li, X. Wang, and X. Wang. A joint syntactic and semantic dependency parsing system based on maximum entropy models. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL 2009): Shared Task*, pages 109–113, Boulder, Colorado, June 2009. Association for Computational Linguistics.

[46] G. van Noord. Using self-trained bilexical preferences to improve disambiguation accuracy. In *Proceedings of the 10th International Conference on Parsing Technologies*, IWPT '07, pages 1–10, Stroudsburg, PA, USA, 2007. Association for Computational Linguistics.

[47] H. Yamada and Y. Matsumoto. Statistical dependency analysis using support vector machines. In *Proceedings of EWPT 2003*, pages 195–206, 2003.

[48] H. Yang and C. Zong. Multi-predicate semantic role labeling. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 363–373, Doha, Qatar, October 2014. Association for Computational Linguistics.

[49] A. Yates, S. Schoenmackers, and O. Etzioni. Detecting parser errors using web-based semantic filters. In *Proceedings of EMNLP 2006*, pages 27–34, 2006.

[50] K. Yu, D. Kawahara, and S. Kurohashi. Cascaded classification for high quality head-modifier pair selection. In *Proceedings of NLP 2008*, pages 1–8, 2008.

[51] B. n. Zapirain, E. Agirre, and L. Màrquez. Generalizing over lexical features: Selectional preferences for semantic role classification. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 73–76, Suntec, Singapore, August 2009. Association for Computational Linguistics.

[52] H. Zhang and R. McDonald. Generalized higher-order dependency parsing with cube pruning. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 320–331, Jeju Island, Korea, July 2012. Association for Computational Linguistics.

[53] H. Zhao, W. Chen, C. Kity, and G. Zhou. Multilingual dependency learning: A huge feature engineering method to semantic dependency parsing. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL 2009): Shared Task*, pages 55–60, Boulder, Colorado, June 2009. Association for Computational Linguistics.

[54] G. Zhou, J. Zhao, K. Liu, and L. Cai. Exploiting web-derived selectional preference to improve statistical dependency parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1556–1565, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

[55] Z.-H. Zhou and M. Li. Tri-training: Exploiting unlabeled data using three classifiers. pages 1529–1541.

List of Publications

[1] Gongye Jin, Daisuke Kawahara, Sadao Kurohashi. Language-independent Approach to High Quality Dependency Selection from Automatic Parses *Journal of Natural Language Processing*, 21(6):1164–1182, 2014.

[2] Gongye Jin, Daisuke Kawahara, Sadao Kurohashi. Chinese Semantic Role Labeling using High-quality Syntactic Knowledge In *Proceedings of The 8th SIGHAN Workshop on Chinese Language Processing (SIGHAN-8)*, pages 120–127, 2015.

[3] Gongye Jin, Daisuke Kawahara, Sadao Kurohashi. A Framework for Compiling High Quality Knowledge Bases From Raw Corpora In *Proceedings of International Conference on Language Resources and Evaluation (LREC)*, pages 109–114, 2014.

[4] Gongye Jin, Daisuke Kawahara, Sadao Kurohashi. High Quality Dependency Selection from Automatic Parses In *Proceedings of International Joint Conference on Natural Language Processing (IJCNLP)*, pages 947-951, 2013.

[5] Gongye Jin, Daisuke Kawahara, Sadao Kurohashi. Language-independent Approach to High Quality Dependency Selection From Automatic Parses In *Proceedings of Information Processing Society of Japan Kansai Branch (IPSJ)*, 2013. (in Japanese).

[6] Gongye Jin, Daisuke Kawahara, Sadao Kurohashi. Selecting High Quality Dependencies from Automatic Parses In *Proceedings of annual meeting of the association for Natural Language Processing (NLP)*, 2013.

[7] Gongye Jin, Daisuke Kawahara, Sadao Kurohashi. A Framework of Automatic Case Frame Construction From Raw Corpus In *Proceedings of annual meeting of the association for Natural Language Processing (NLP)*, 2012.

[8] Gongye Jin, Daisuke Kawahara, Sadao Kurohashi. Automatic Construction of Multilingual Case Frames In *Proceedings of Information Processing Society of Japan Kansai Branch (IPSJ)*, 2011.

As a general rule, the copyright of a paper approved for publication belongs to the Association. (The copyright must be transferred to the Association at the time of the submission of the final manuscript for printing.) If the copyright belongs to an affiliated organization or other party, making it difficult for the Association to own the copyright, the matter will be discussed upon request. Please note that the Association may make published papers electronically accessible through J-STAGE, an online article database service, and other services, so that the findings presented in the papers may be widely shared and may contribute to future academic research. If the author of the published paper digitizes such paper and releases it to third parties using digital media such as computer networks or CD-ROMs, the volume, number, and pages of the Journal of Natural Language Processing of the publication must be indicated in a clear manner for all viewers.