國立臺灣大學電機資訊學院電子工程學研究所
碩士論文
Graduate Institution of Electronics Engineering
College of Electrical Engineering and Computer Science
National Taiwan University
Master Thesis

以正規邏輯方法解決中文文本蘊含辨識問題
A Formal Logic Approach to Chinese Recognizing Textual Entailment

張富傑
Fu-Chieh Chang

指導教授：黃鐘揚博士
Advisor: Chung-Yang(Ric) Huang

中華民國 103 年 7 月
July, 2014

誌謝

經過兩年的碩士班，最後終於能完成這篇碩士論文，首先我要感謝指導教授黃鐘揚老師，給予我這個頗有挑戰性的跨領域題目，也感謝黃鐘揚老師教我程式設計以及邏輯驗證的相關知識，還有幫助我做最後的論文修定。

另外，關於完成這篇論文所需的相關知識及技術，要感謝的人太多了。感謝教我課堂上的相關知識，或者提供研究資源，以及給予意見者，包括：臺大資管系蔡益坤教授，臺大語言學研究所謝舒凱教授，臺大電機系陳銘憲教授，臺大資工系林軒田教授、陳信希教授，中正大學語言學研究所吳俊雄教授，意藍科技董事總經理楊立偉博士，卓騰語言科技公司的創辦人王文傑先生，Storysense Computing, Inc. 的創辦人沈育德先生、共同創辦人黃鼎祥、林挺正先生，以及 VisualOn, Inc. 的謝政達博士等人。

最後，感謝 DVLab 實驗室的學長姐、同學以及學弟，包括：Jason、Gary、Rocky、穎哥、賴建宇、碰皮、朱弟、芸貓、倪鈴雅、王冠璋、Krista、Nanny、王宏偉、小熊、范寬、江弘勛、Calvin ... 等人，一起度過實驗室中的研究及玩樂的生活。還有，感謝女朋友 Silke，以及家人的陪伴，支持與鼓勵。

摘要

在自然語言處理的應用中，理解自然語言，一直是個很有挑戰的問題。傳統的自然語言處理研究，著重在理解語言的語意與邏輯。而目前自然語言處理的的研究方向，則是著重在用巨量資料和機器學習的方式。雖然這兩種方法各有優缺點，但現今在自然語言處理的研究，傳統的語意學模型則極少被拿出來討論。而目前的機器學習方法，也有其解決問題的極限。若能整合傳統的語意學，和機器學習的方法，是一個值得研究的方向。

我們建構一個系統可以用正規邏輯方法解決中文文本蘊含辨識問題。基於形式語意學和計算語意學的理論，我們先用機器學習的方式，將中文文句轉成剖析樹，再用我們提出的演算法，把剖析樹轉成語意表達式。並且，我們提出可以整合外部的知識和語意表達式的方法，並用定理證明的方式，解決中文文本蘊含辨識的問題。再來我們示範，我們的系統可以解決句型較簡單的問題。以及解決現實世界應用問題的可能性與挑戰。最後，我們得出這個系統的優缺點，以及未來可行的研究方向，來改進此系統。

Abstract

In the research of natural language processing (NLP), understanding the natural language is always a challenging problem. Traditionally, the research of NLP focuses on the semantics and logic of natural language. However, the present NLP research trend is focusing on the big data and machine learning techniques. These two methods have their own pros and cons; however, the traditional research of semantics and logic are seldom discussed in the recent works, and the existing machine learning techniques also have their limitations. Combining the traditional works on semantics with machine learning techniques is a good perspective to research.

We build a system to solve the Chinese recognizing textual entailment (RTE) problem by formal logic method. Based on the theory of formal semantics and computational semantics, first, we use the machine learning technique to convert Chinese sentences in natural language into syntax trees. Then, we propose an algorithm to convert the syntax trees into semantic representations. Also, we propose a method that solves the RTE problem by integrating external knowledge resources with the proposed semantic representations. With these semantic representations, we can use the theorem proving techniques to solve the problem of Chinese RTE. Then, we demonstrate that our approach can solve some simple cases of Chinese RTE. Also, we show the possibilities and difficulties to solve the real-world cases. Finally, we point out the strengths and weaknesses of our system, and the possibilities on future research to improve our system.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this research, we try to find a method that makes computer understand the meaning of text in natural language.

Natural Language Understanding (NLU) is the study of how computer can understand natural language. It is a challenging task of Natural Language Processing (NLP). It requires the study of semantics, the study of meaning, to represent the meaning of natural language. One approach to represent the meaning is to represent the meaning by formal logic, and this approach is called formal semantics. If the sentences in natural language are represented in logical forms, computer can analyze the truth condition of these sentences, or draw inferences according to these sentences. Recognize textual entailment (RTE) is a task of natural language understanding. It is the task that given two text fragments, deciding whether the meaning of one text is entailed from the other. If both of the text fragments are converted into logical forms, we can easily decide whether one logical formula is entailed from the other.

In recent years, with the advance of machine learning techniques, it becomes possible for computers to learn how to build the logical forms automatically. However, the process of building logical forms is susceptible to the errors introduced by machine learning algorithms. If the error exists in the logical form, the computer will misunderstand the meaning. Also, the advance of machine learning approaches demonstrates that it is possible to solve the task of NLP without understanding the meaning in natural language. For example, by machine learning approach, the problem of RTE can be solved only by analyzing the surface similarities, such as lexical similarity and syntactic similarity, between two text fragments, without analyzing the deep semantics of the sentences. To draw an analogy, a student can pass an exam if he practices lots of exercises in textbooks without understanding the content of textbooks. However, a student can't pass an exam if he misunderstands the content of textbooks. Hence, in real-world applications, machine learning approaches usually perform better than traditional NLU approaches. After 1990s, the research trends of NLP are focusing on machine learning approaches, leading to a decrease in the works of NLU.

However, some tasks in NLP, such as document summarization, question answering and RTE, can't be solved with high precision without understanding the meaning of natural language. In recent years, there is a resurgence of research on NLU, mainly using the techniques from previous decades, such as formal logic. However, most of these works care only about English, and there are only few works of computational semantics focusing on Chinese.

In this thesis, we do not intend to show that our formal logic approach is better than (or just able to compete with) existing machine learning approaches in the Chinese RTE task, rather than we want to show the possibilities and difficulties to solve the Chinese RTE task by formal logic approach, with the underlying spirit of NLU.

1.1 Contributions of this Thesis

In this thesis, we build a system that can automatically construct the semantic representation in logical form from natural language in Chinese, and we use the RTE tasks to evaluate the capability of our system. We point out the possibilities and difficulties of solving RTE tasks by our method, and our system can be served as a framework for future research on computational semantics and NLU. Our contributions can be categorized into three parts: for Chinese computational semantics, for Chinese RTE and for future research.

1.1.1 For Chinese Computational Semantics

We present an algorithm to build semantic representations from the syntax trees produced by the CKIP Chinese Parser. By this algorithm, we can build semantic representation from free text. These semantic representations can cover several linguistic properties, such as negation and conjunction.

1.1.2 For Chinese RTE

We present a formal logic approach to solve the Chinese RTE problem by using the theorem prover to prove the satisfiability of the logical form. We also present a simple algorithm to construct the logical form of missing knowledge required to solve RTE according to the input logical form, and we use some external knowledge resources to validate the correctness of the logical form of missing knowledge. Then, we demonstrate that our formal logic approach can solve the Chinese RTE tasks of simple sentences. We also show the possibilities and difficulties to solve the Chinese RTE tasks of real-world problems.

1.1.3 For Future Research

We build a framework that integrates the whole NLP pipeline from Chinese word segmentation to semantic representation. It also integrates some external knowledge resources such as the Chinese WordNet and the Chinese Latent Semantic Analysis (LSA) Website. This framework provides a good environment for future research on Chinese computational semantics and Chinese RTE.

1.2 Organization of this Thesis

First, we introduce the required background knowledge, such as formal semantics, knowledge resources and RTE problem in Chapter 2, and we give a brief review on related research about computational semantics and RTE in Chapter 3. In Chapter 4, we elaborate on the implementation of our system and show a detailed illustration of our system architecture and algorithm. In Chapter 5, we discuss the weaknesses and limitations of our implementation. In Chapter 6, we demonstrate the capabilities and limitations of our system in solving simple RTE cases and real-world RTE cases. In Chapter 7, we conclude this research and make a comparison to other previous research. In Chapter 8, we propose several topics on how this system can be improved in the future research, and the possibility of integrating this system into some real-world applications.

Chapter 2

Preliminaries

In this chapter, we introduce some required background knowledge of our work, including natural language understanding, semantics, theorem proving, RTE and knowledge resources. Natural language understanding is a subfield of natural language processing. In order to achieve the understanding of natural language, we need to study how to represent the meaning as logical form. Formal semantics is the study of expressing the meaning of natural language by logical form. Computational semantics is the technique about how to automatically convert sentence in natural language into logical form. Theorem proving is the technique to prove whether a logical statement is true or false. RTE is a task to recognize whether a text T entails a hypothesis H. Knowledge resources are the resources of lexical knowledge and world knowledge that can be supplied to help the NLP tasks.

2.1 Natural Language Understanding

Natural language understanding (NLU) [1, 2] is a subfield of natural language processing (NLP). The characteristics of NLU systems are that they deal with the machine reading comprehension. The goal of an NLU system is to interpret an input text fragment. The process of interpretation can be viewed as a translation from a natural language text fragment into a representation in an unambiguous formal language, such as formal logic. Using formal logic to represent text fragments in natural language has the following two properties: First, the representation is precise and unambiguous. Second, the representation captures the intuitive structure of the natural language sentences that it represents. Beside semantic representation, many researchers have come to the conclusion that linguistic knowledge and world knowledge play important roles in understanding the natural language. Finally, if we want to demonstrate the capability of an NLU system, it can be evaluated by the task of RTE. In this chapter, we will introduce the concepts of formal logic representation, knowledge resources and RTE in the following sections.

2.2 Formal Semantics

Formal semantics is the study of representing the meaning of natural language by formal logic, mainly first-order logic. For example, sentence (2.1) can be converted into logical form as (2.2).

$$Brutus\ stabbed\ Ceasar. \tag{2.1}$$

$$stab(Brutus, Ceasar) \tag{2.2}$$

In other words, the sentence (2.1) can be characterized as a predicate-argument structure that maps sentences onto logical forms. In this structure, the verb $stab$ is a predicate, the subject of the verb $Brutus$ is the first argument, and object of the verb $Ceasar$ is the second argument. Although this type of representation can express the relationships between verb, subject and object, however, it has some drawbacks. For example, if we add a prepositional phrase (PP) $with\ knife$ into (2.1), as (2.3). If we want to convert (2.3) into semantic representation by this procedure, it will be transformed into (2.4).

$$Brutus\ stabbed\ Ceasar\ with\ knife. \tag{2.3}$$

$$stab(Brutus, Ceasar, knife) \tag{2.4}$$

In (2.4), we can see that the third argument $knife$ is the instrument of the verb $stab$. However, if we add another PP $in\ the\ agora$ into (2.1), it will become (2.5).

$$Brutus\ stabbed\ Ceasar\ in\ the\ agora. \tag{2.5}$$

By this procedure, it will be transformed into logical form (2.6).

$$stab(Brutus, Ceasar, agora) \tag{2.6}$$

Obviously, the third argument in (2.6), $agora$, is a location. However, in (2.4), the third argument is an instrument, and hence the inconsistency of the type in the third argument arises.

2.2.1 Davidsonian Event Semantics

Since we do not know that how many PPs can exist in a sentence. It is impossible to allocate the argument slots of a verb predicate for PPs. Davidson [3] introduced the notion of *event* as quoted below.

Adverbial modification is thus seen to be logically on a par with adjectival modification: what adverbial clauses modify is not verbs but the events that certain verbs introduce.

It means that the phrases that modify a verb actually modify the event introduced by that verb. With this in mind, we can convert the sentences (2.1), (2.3) and (2.5) into logical forms (2.7), (2.8) and (2.9), respectively.

$$\exists e.stab(Brutus, Ceasar, e) \tag{2.7}$$

$$\exists e.stab(Brutus, Ceasar, e) \wedge instrument(knife, e) \tag{2.8}$$

$$\exists e.stab(Brutus, Ceasar, e) \wedge location(agora, e) \tag{2.9}$$

In the above equations, the roles of PPs, such as *instrument* and *location*, become predicates. The variable e is the event of the verb, and it serves as the bridges between the verb and other modifiers such as PP. By this representation, we can add any number of PPs without changing the counts of argument slots of the verb predicate. However, another problem arises when some arguments are missing from the verb predicate. For example, in (2.10), the object of the verb is missing. After we convert it into logical form as shown in (2.11), we do not know how to fill in the slot for the second argument in $stab$.

$$Brutus\ stabbed\ unexpectedly. \tag{2.10}$$

$$\exists e.stab(Brutus,\ ,e) \wedge unexpectedly(e) \tag{2.11}$$

2.2.2 Neo-Davidsonian Semantics

It is possible that the subject, object or other arguments of a verb are absent. The solution is that we can treat all subjects and objects as modifiers of the verb. For instance, we can transform the sentence (2.1) into logical form (2.12). Undoubtedly, the sentence (2.10) can be transformed into logical form (2.13) without any unfilled argument slot.

$$\exists e.stab(e) \wedge subject(Brutus, e) \wedge object(Ceasar, e) \tag{2.12}$$

$$\exists e.stab(e) \wedge subject(Brutus, e) \wedge unexpectedly(e) \tag{2.13}$$

With the concept of *thematic role* [4], which describes the relationship between noun and verb, we assign the role *agent* to subject of verb $stab$, and the role *patient* to object. Then, the logical forms (2.12) and (2.13) become (2.14) and (2.15), respectively.

$$\exists e.stab(e) \wedge agent(Brutus, e) \wedge patient(Ceasar, e) \tag{2.14}$$

$$\exists e.stab(e) \wedge agent(Brutus, e) \wedge unexpectedly(e) \tag{2.15}$$

This kind of representation is called Neo-Davidson Semantics [5].

2.3 Computational Semantics

After knowing how to convert a sentence into its semantic representation, the question is: *How can we automatically convert sentences into semantic representations by com-*

puter program? This question is solved by the study of computational semantics. We can use *Lambda Calculus* to build the semantic representation [6]. For example, if we want to convert the sentence (2.1) into logical form (2.2), the first step is to convert it into syntax tree, as illustrated in Figure 2.1. Then, we can construct the semantic representation by the procedure illustrated in Figure 2.2.

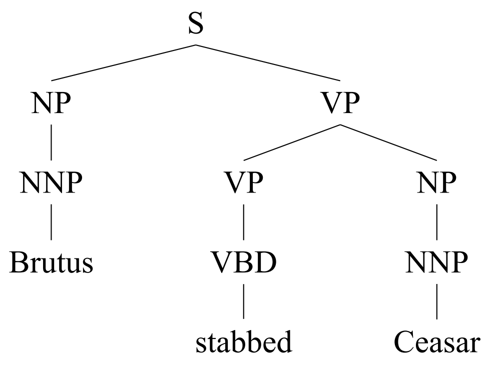


Figure 2.1: Example of Syntax Tree

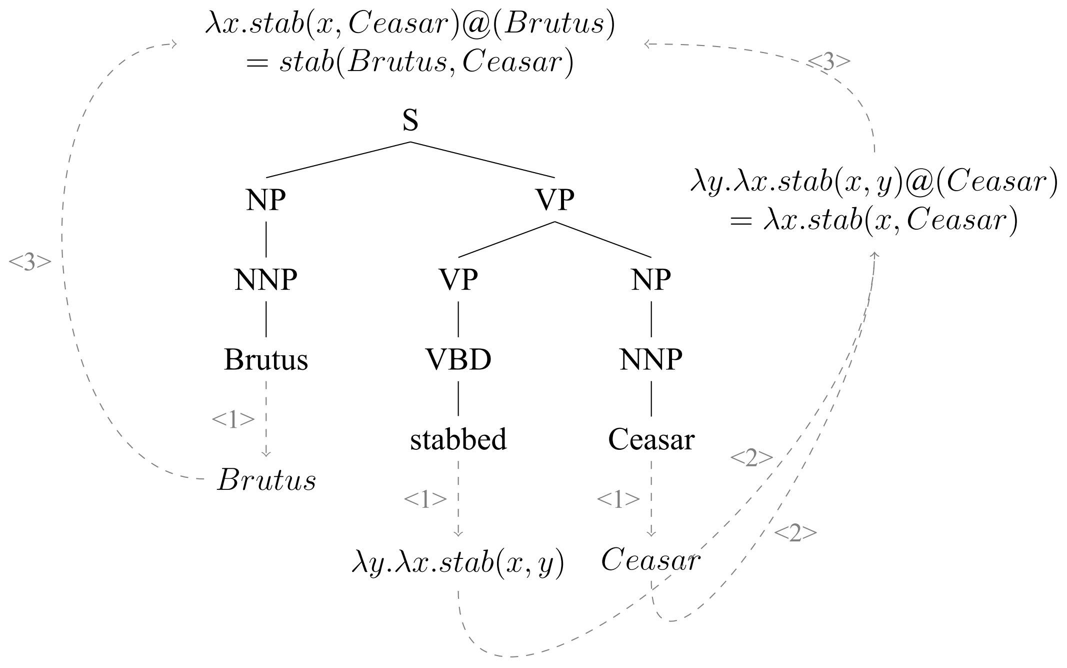

$\lambda x.stab(x, Ceasar)@(Brutus)$
$= stab(Brutus, Ceasar)$
<3>
S
NP
VP
$\lambda y.\lambda x.stab(x, y)@(Ceasar)$
$= \lambda x.stab(x, Ceasar)$
<3>
NNP
VP
NP
Brutus
VBD
NNP
<1>
stabbed
Ceasar
<2>
$Brutus$
<1>
<1>
<2>
$\lambda y.\lambda x.stab(x, y)$
$Ceasar$

Figure 2.2: Procedure of Semantics Construction

< 1 > Define the semantic representation for the verb $stab$ as $\lambda y.\lambda x.stab(x, y)$. Also, the semantic representations for the proper nouns Brutus and Ceasar are $Brutus$ and

$Ceasar$, respectively.

< 2 > Combine the semantic representations according to the structure of this syntax tree. First, combine the semantic representations $\lambda y.\lambda x.stab(x, y)$ and $Ceasar$ by function application: $\lambda y.\lambda x.stab(x, y)@(Ceasar)$. The symbol @ denotes function application. The resulted semantic representation is $\lambda x.stab(x, Ceasar)$.

< 3 > Combine the semantic representations $\lambda x.stab(x, Ceasar)$ and $Brutus$ by function application: $\lambda x.stab(x, Ceasar)@(Brutus)$. Finally, we get the semantic representation of (2.2), $stab(Brutus, Ceasar)$, and it is successfully constructed by this procedure.

By this procedure, we can also construct the semantic representations introduced in Section 2.2.1 and Section 2.2.2.

2.4 Automated Theorem Proving

Automated Theorem Proving (ATP) is the study of how computer programs can prove that whether a logical statement is true or false. One of the methods for ATP is *refutation*. This method is to negate the logical statement supposed to be proved (the conclusion), and then add it to the list of premises. For example, given two logical formulas in (2.16), the premise is P and the conclusion is Q.

$$\begin{aligned} P &= A(x_1) \wedge B(x_2) \wedge C(x_3) \\ Q &= A(x_1) \wedge B(x_2) \end{aligned} \tag{2.16}$$

If we want to prove that $P \rightarrow Q$ is true, we can use a theorem prover to prove whether $P \wedge \neg Q$, (2.17), is satisfiable or not.

$$\begin{aligned} & P \wedge \neg Q \\ &= A(x_1) \wedge B(x_2) \wedge C(x_3) \wedge \neg(A(x_1) \wedge B(x_2)) \\ &= A(x_1) \wedge B(x_2) \wedge C(x_3) \wedge (\neg A(x_1) \vee \neg B(x_2)) \end{aligned} \tag{2.17}$$

If (2.17) is unsatisfiable, $P \rightarrow Q$ is true; otherwise, $P \rightarrow Q$ is false. *Tableaux* and *Resolution* are two main algorithms that can solve the problem of ATP.

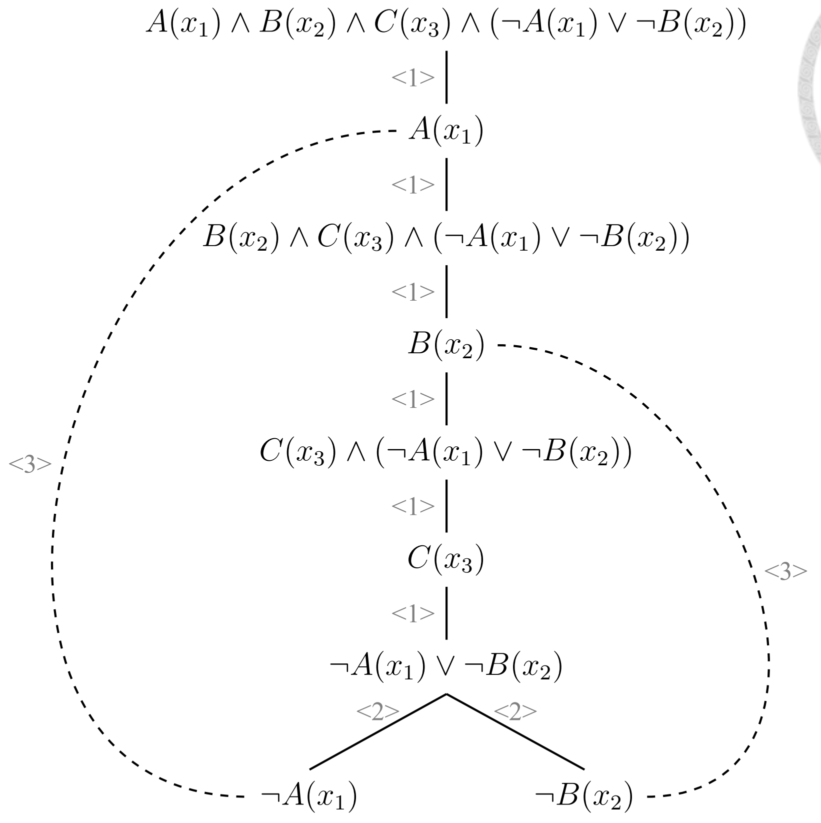
2.4.1 Tableaux

The goal of tableaux algorithm is to show that whether a formula is satisfiable or not. The main idea of tableaux algorithm is to break down a complex formula into several atomic formulas, until complementary pairs of atomic formulas are produced or no further possible break-down exists. This method works on a tree (tableau) with nodes labeled with formulas. At each step, we modify the tree by adding a descendant to a leaf in the tree. For example, if we want to prove (2.17) is unsatisfiable, we initialize the root of tree as $P \wedge \neg Q$. Then, we decompose the formula at the root of tree and check the satisfiability by the following procedures, shown in Figure 2.3a.

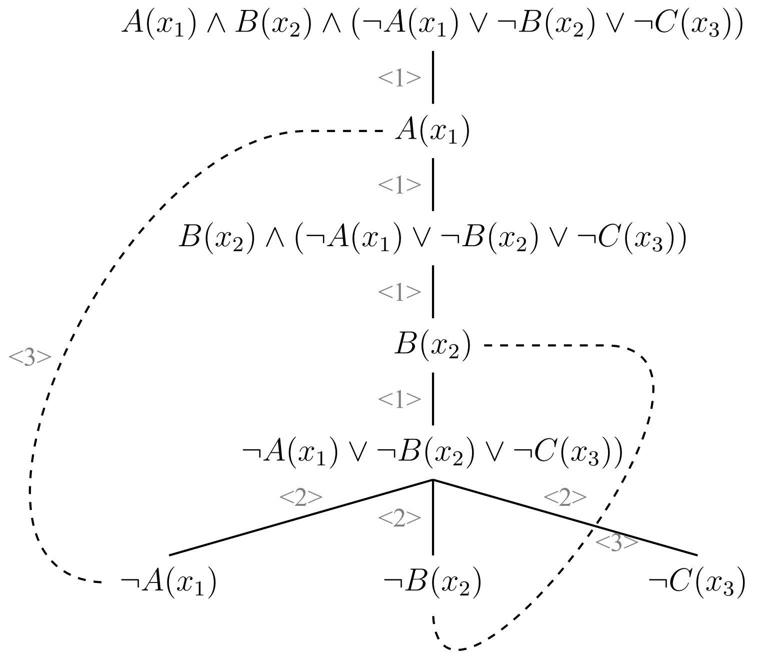
< 1 > For every conjunction of formula, such as $A \wedge B$, we decompose it into two formulas, A and B, and append A to the node with formula $A \wedge B$ first, and then append B to the node with formula A. In Figure 2.3a, we decompose the formula in the root, $A(x_1) \wedge B(x_2) \wedge C(x_3) \wedge (\neg A(x_1) \vee \neg B(x_2))$, into $A(x_1)$ and $B(x_2) \wedge C(x_3) \wedge (\neg A(x_1) \vee \neg B(x_2))$. We append $A(x_1)$ to the root of the tree first, and then append $B(x_2) \wedge C(x_3) \wedge (\neg A(x_1) \vee \neg B(x_2))$ to the node with formula $A(x_1)$. Since $B(x_2) \wedge C(x_3) \wedge (\neg A(x_1) \vee \neg B(x_2))$ can be further decompose into $B(x_2)$ and $C(x_3) \wedge (\neg A(x_1) \vee \neg B(x_2))$, we decompose it and do the same procedure again, until no conjunction can be decomposed.

< 2 > For every disjunction of formula, such as $A \vee B$, we decompose it into two formulas, A and B, and then append both of them to the node with formula $A \vee B$. In Figure 2.3a, we decompose the formula $\neg A(x_1) \vee \neg B(x_2)$ into $\neg A(x_1)$ and $\neg B(x_2)$, and then append both of them to the node with formula $\neg A(x_1) \vee \neg B(x_2)$.

< 3 > If there is no further possible decomposition, we check the satisfiability of this tableau by the following procedure. For every path from the root to the leaf of this tableau, if a path contains both A and $\neg A$, and A is an atomic formula, this


$A(x_1) \wedge B(x_2) \wedge C(x_3) \wedge (\neg A(x_1) \vee \neg B(x_2))$
<1>
$A(x_1)$
<1>
$B(x_2) \wedge C(x_3) \wedge (\neg A(x_1) \vee \neg B(x_2))$
<1>
$B(x_2)$
<1>
<3>
$C(x_3) \wedge (\neg A(x_1) \vee \neg B(x_2))$
<1>
$C(x_3)$
<3>
<1>
$\neg A(x_1) \vee \neg B(x_2)$
<2>
<2>
$\neg A(x_1)$
$\neg B(x_2)$

(a) $P \wedge \neg Q$ is Unsatisfiable


$A(x_1) \wedge B(x_2) \wedge (\neg A(x_1) \vee \neg B(x_2) \vee \neg C(x_3))$
<1>
$A(x_1)$
<1>
$B(x_2) \wedge (\neg A(x_1) \vee \neg B(x_2) \vee \neg C(x_3))$
<1>
<3>
$B(x_2)$
<1>
$\neg A(x_1) \vee \neg B(x_2) \vee \neg C(x_3))$
<2>
<2>
<2>
<3>
$\neg A(x_1)$
$\neg B(x_2)$
$\neg C(x_3)$

(b) $P \wedge \neg Q$ is Satisfiable

Figure 2.3: Procedure of Tableaux Algorithm

path is *closed*. If all the paths in the tableau are closed, this tableau is unsatisfiable; otherwise, it is satisfiable. In Figure 2.3a, the path from the root to the node with formula $\neg A(x_1)$ is closed, because it contains both $\neg A(x_1)$ and $A(x_1)$, annotated by the dashed line. Also, the path from the root to the node with formula $\neg B(x_2)$ is closed for the same reason. Since all the paths in this tableau are closed, this tableau is unsatisfiable. If we exchange P and Q, as shown in Figure 2.3b, the path from the root to the node with formula $\neg C(x_3)$ is not closed, because it does not contain both $\neg C(x_3)$ and $C(x_3)$. Since this tableau contains a non-closed branch, it is satisfiable.

2.4.2 Resolution

Resolution is an inference technique, and it produces a new clause by combining two clauses containing complementary atomic formulas. For example, given two clauses, $A(x) \vee B(x)$ and $\neg A(x)$, we can derive the conclusion $B(x)$ by the resolution rule shown in (2.18).

$$\frac{A(x) \vee B(x), \neg A(x)}{B(x)} \tag{2.18}$$

We can use the resolution rule to prove that (2.17) is unsatisfiable, and the procedure is presented in Figure 2.4a.

In 1-3, since P is conjunctions of atomic formulas, we decompose P into three formulas.

In 4, since $\neg Q$ is disjunctions of atomic formulas, we do not need to decompose it.

In 5, we use the rule of resolution with 1 and 4 to eliminate the atomic formula $\neg A(x_1)$ in 4.

In 6, we eliminate $\neg B(x_2)$ in 5 by resolution, and we get an empty clause.

After these procedures, if an empty clause can be generated, it means that $P \wedge \neg Q$ is unsatisfiable. On the other hand, if we exchange P and Q, as illustrated in Figure 2.4b, we will leave a clause $\neg C(x_3)$ at the end of this procedure. The clause $\neg C(x_3)$ can't be

1.	$\{A(x_1)\}$	
2.	$\{B(x_2)\}$	
3.	$\{C(x_3)\}$	
4.	$\{\neg A(x_1), \neg B(x_2)\}$	
5.	$\{\neg B(x_2)\}$	resolve(1, 4)
6.	$\{\}$	resolve(2, 5)

(a) $P \wedge \neg Q$ is Unsatisfiable

1.	$\{A(x_1)\}$	
2.	$\{B(x_2)\}$	
3.	$\{\neg A(x_1), \neg B(x_2), \neg C(x_3)\}$	
4.	$\{\neg B(x_2), \neg C(x_3)\}$	resolve(1, 3)
5.	$\{\neg C(x_3)\}$	resolve(2, 4)

(b) $P \wedge \neg Q$ is Satisfiable

Figure 2.4: Procedure of Resolution Algorithm

eliminated by the rule of resolution. As a result, an empty clause can't be derived; hence $P \wedge \neg Q$ is satisfiable.

2.5 Recognizing Textual Entailment

The Recognizing Textual Entailment (RTE) task is to recognize, given two text fragments, whether the meaning of one text can be inferred (entailed) from the other [7]. Given a pair of text expressions, denoted by T, the entailing Text, and H, the entailed *Hypothesis*, if the meaning of H can be inferred from the meaning of T, we say that T entails H. For instance, in the text pair T in (2.19), and H in (2.20), we can infer the meaning of H in (2.20) from the meaning of T in (2.19).

$$T \ : \ Jones\ buttered\ the\ toast\ in\ the\ bathroom\ at\ midnight. \tag{2.19}$$

$$H \ : \ Jones\ buttered\ the\ toast. \tag{2.20}$$

In some cases, background knowledge is required to complete the inference task in RTE. For example, in the text pair T (2.19) and H (2.21), if we want to infer the meaning of H in (2.21) from the meaning of T in (2.19), we need to know the fact that *at night* can be inferred from *at midnight*.

$$H \ : \ Jones\ buttered\ the\ toast\ at\ night. \tag{2.21}$$

RTE is a challenging task in natural language processing, and the algorithms to solve RTE can be categorized into two types: formal logic approach and machine learning approach.

2.5.1 Formal Logic Approach

Formal Logic Approach is to convert the sentences into logical forms, and then use theorem prover to prove the truth of entailment [8]. For example, if we want to prove the entailment in the text pair T (2.19) and H (2.20), we need to convert the sentences (2.19) and (2.20) into logical forms in (2.22) and (2.23), respectively.

$$\begin{gathered} \exists e.agent(Jones, e) \wedge buttered(e) \wedge theme(toast, e) \\ \wedge location(bathroom, e) \wedge time(midnight, e). \end{gathered} \tag{2.22}$$

$$\exists e.agent(Jones, e) \wedge buttered(e) \wedge theme(toast, e). \tag{2.23}$$

Then, we can use a theorem prover to prove that (2.22) entails (2.23) is true; hence, H (2.20) can be inferred from T (2.19). To solve the cases that requires background knowledge, such as the text pair T (2.19) and H (2.21), we can convert the sentence (2.21) into logical form (2.24), and the required knowledge that event happened *at night* can be inferred from event happened *at midnight* can be converted into logical form (2.25), too.

$$\exists e.agent(Jones, e) \wedge buttered(e) \wedge theme(toast, e) \wedge time(night, e). \tag{2.24}$$

$$\forall e.time(midnight, e) \rightarrow time(night, e) \tag{2.25}$$

Then, the theorem prover can prove that (2.22) $\wedge$ (2.25) $\rightarrow$ (2.24) is true, so that T (2.19) entails H (2.21) is true.

Since the logical forms need to be automatically constructed by the procedure introduced in Section 2.3, first, we need to build the syntax tree from raw text. However, using the existing parsing algorithm, such as statistical parsing, is possible to produce the erroneous syntax tree, and the error of syntax tree usually propagates to the error of logical form. Hence, this approach is sensitive to the error produced during the preceding stages.

2.5.2 Machine Learning Approach

The RTE task can be viewed as a binary classification problem of classifying whether T entails H is true or false. Machine learning, mainly supervised machine learning, including decision tree[9], support vector machine (SVM) [10], and so on, is a technique to solve this kind of classification problem. The overall process of supervised classification is presented in Figure 2.5 [11].

a. Training Phase

In the Training Phase, the input data consists of two parts, *text* and *label*, as shown in Figure 2.6a. Text contains pairs of T and H, and label is the truth of whether T entails H, manually annotated by human labor. Then, the *features* of text are extracted from text by *feature extractor*. These features can be anything that can be used to distinguish the difference between whether T entails H or not. There are a wide variety of features to choose, such as word similarity, syntactic similarity, tree edit distance, existence of negation word, difference in numerical words, etc. The choices of features can have a huge impact on the precision of classifier. The result after feature extraction is a matrix of features with corresponding numerical values, as shown in Figure 2.7a. Then, the *machine learning algorithm* can train a *classifier model* according to the labels and the features.

b. Prediction Phase

In the Prediction Phase, the input data consists of text, without the label of truth of entailment, as illustrated in Figure 2.6b. First, we use the feature extractor to extract the features from text, similar to the matrix presented in Figure 2.7a. Then, we can use the classifier model trained by machine learning algorithm to predict the values of labels.

Based on the values of features, the classifier model can generate the result of truth of whether T entails H, as shown in Figure 2.7b.

Unlike formal logic method, the performance of machine learning method will not be directly affected by the error during building syntax tree; nonetheless, supervised machine learning method requires huge amount of human labor in annotating data for training.

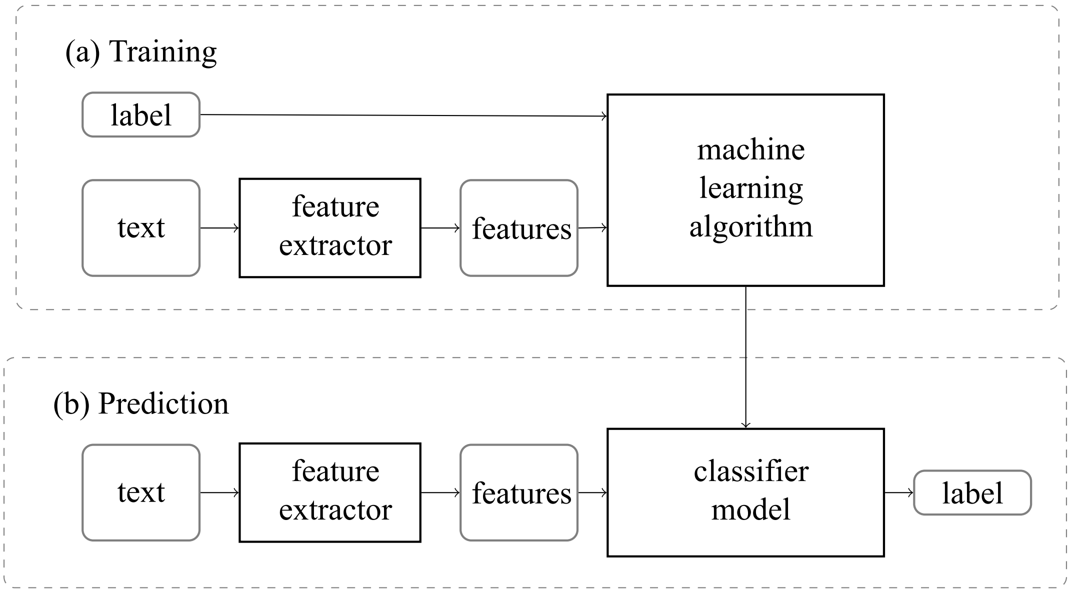


Figure 2.5: Procedure of Supervised Machine Learning for Classification

id	text	label
1	T : An American won the Nobel prize for literature. H : An American won the Nobel prize.	True
2	T: The Council of Europe has 45 member states. H: The Council of Europe is made up by 45 member states.	True
3	T: Jody saw Gary sign the contract. H: Judy saw Gary sign the contract and his secretary make a copy.	False
...	....	...

(a) Input Data in Training Phase

id	text
1	T : Einstein won the Nobel prize for physics. H : Einstein won the Nobel prize.
2	T: The Council of Europe has 45 member states. H: The Council of Europe is made up by 42 member states.
3	T: Nancy saw Steven playing the piano in the living room. H: Nancy saw Steven playing the piano.
...	....

(b) Input Data in Prediction Phase

Figure 2.6: Input Data for Machine Learning

id	$feature_1$	$feature_2$	$feature_3$	...
1	2.0000	3.5001	1.3319	...
2	-3.0000	2.4232	2.1121	...
3	-6.0000	-1.7121	1.0023	...
...	...	...	...	...

(a) Matrix of Feature Values

id	*label*
1	True
2	False
3	True
...	...

(b) Result of Prediction

Figure 2.7: Feature Values and Results

2.6 Knowledge Resources

In RTE task, extra knowledge is usually required to solve the entailment problem. Resources of knowledge can be categorized into two types: lexical knowledge (the meaning of words) and world knowledge (general concept about the world). Lexical knowledge and world knowledge can be created by human labor, such as lexical semantics and ontology. Knowledge can also be extracted automatically from corpus, such as distributional semantics.

2.6.1 Lexical Semantics

Lexical knowledge can be created by the study of lexical semantics. Lexical knowledge provides the knowledge of meaning of words and the relation between words. WordNet [12] is a lexical knowledge resource for English. It groups words (*lemmas*) into sets of *synonyms*, called *synset*, based on their meanings, and provides semantic relations between these synsets. There are several types of semantic relations, including:

- *hypernyms* : Y is a *hypernym* of X if every X is a Y (canine is a hypernym of dog)
- *hyponyms* : Y is a *hyponym* of X if every Y is a X (dog is a hyponym of canine)
- *meronym* : Y is a *meronym* of X if Y is a part of X (tree is a meronym of forest)
- *holonym* : Y is a *holonym* of X if X is a part of Y (forest is a holonym of tree)

For instance, the word *dog*, is in the synset $Synset(dog)$, and this synset has three lemmas, including $Lemma(dog)$, $Lemma(domestic\_dog)$ and $Lemma(Canis\_familiaris)$. The hypernyms of $Synset(dog)$ are $Synset(domestic\_animal)$ and $Synset(canine)$, and the hyponyms of $Synset(dog)$ are $Synset(puppy)$, $Synset(pooch)$, ..., and $Synset(griffon)$, etc. These semantic relations are illustrated in Figure 2.8.

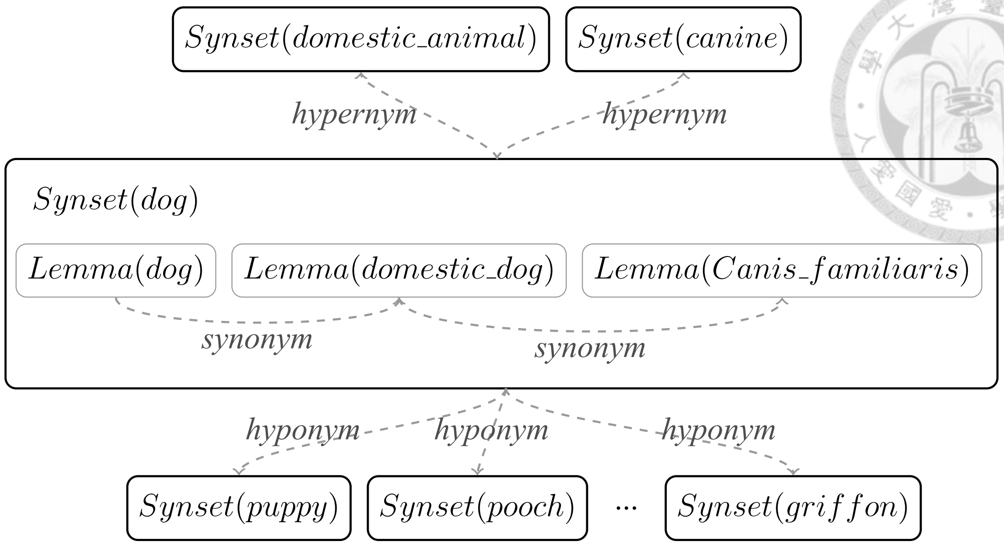


Figure 2.8: Example of Semantic Relations in WordNet

VerbNet [13] is another type of lexical knowledge resource. It provides the link between syntax and semantics of verbs. Each VerbNet class contains a set of verbs with a list of thematic roles, such as agent, theme and location, etc. Several syntactic frames are assigned to each verb class. Syntactic frames describe all possible surface realizations of the verbs. For example, the verb class $eat - 39.1 - 1$ has four syntactic frames, and one of them is $NP\ \ V\ \ NP$, and the corresponding thematic roles are $Agent\ \ V\ \ Patient$. All the syntactic frames of verb class $eat - 39.1 - 1$ are presented in Table 2.1.

syntactic frame	thematic role	example
NP V NP	Agent V Patient	Cynthia ate the peach.
NP V	Agent V	Cynthia ate.
NP V PP-Conative	Agent V (at) Patient	Cynthia ate at the peach.
NP V PP.source	Agent V ((+src)) Source	He ate off of the table.

Table 2.1: Example of Syntactic Frame in VerbNet

2.6.2 Ontology

World Knowledge is the knowledge that is independent from language. For example, the knowledge that *a bicycle has two wheels* is an example of world knowledge. No matter what kind of language we choose to describe it, the concept of this knowledge remains the same. However, a problem arises. How can we formally represent the world knowledge? The solution is to represent it by ontology, which formally represents world knowledge as a hierarchy of concepts and interrelationships between these concepts [14]. For example, if we want to describe the world knowledge in Figure 2.9, we can use ontology to conceptualize this knowledge, as illustrated in Figure 2.10. In this figure, the circular nodes represent the concepts, and the arrow-headed lines represent the relationships between concepts. For instance, the concept *Car* is a subclass of the concept *Vehicle*, and its relationship between *Engines* is *has*, and the word *min* represents the cardinality of relationship. Ontology can also be represented by logical form, such as description logic [15]. For example, we can represent the ontology in Figure 2.10 by description logic, as shown in (2.26). In this formula, the symbol $\sqsubseteq$ means subset $\subseteq$, the symbol $\sqcap$ means intersection $\cap$, the symbol $\geq_3$ means the cardinality is greater or equal to 3, and the symbol $=_2$ means the cardinality is exactly equal to 2.

$$\begin{aligned} &Car \sqsubseteq Vehicle \sqcap (\geq_3 has.Wheel) \sqcap (\geq_1 has.Engine) \\ &Bicycle \sqsubseteq Vehicle \sqcap (=_2 has.Wheel) \end{aligned} \tag{2.26}$$

A car has more than two wheels.
A car has one or more engines.
Car is a kind of vehicle.
A bicycle has two wheels.
Bicycle is a kind of vehicle.

Figure 2.9: Example of World Knowledge

2.6.3 Distributional Semantics

Unlike lexical semantics and ontology that require human labor to build, distributional semantics can be automatically built by unsupervised machine learning technique. The basic idea of distributional semantics is based on the hypothesis: *Words which are similar in meaning occur in similar contexts* [16]. There is a wide variety of computational models implementing distributional semantics, including Latent Semantic Analysis (LSA) [17], Hyperspace Analogue to Language (HAL) [18], syntax-based or dependency-based mod-

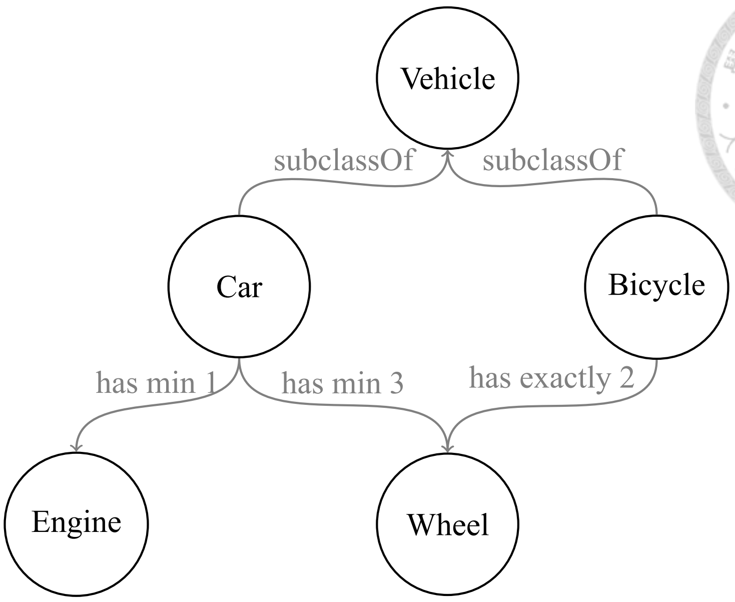


Figure 2.10: Example of Ontology

els, and so on. In this work, we focus on LSA, which is a statistical technique to measure the semantic similarity between words. The overall procedure of LSA is demonstrated in Figure 2.11. This technique converts words into vectors, and measure the similarity between two words by calculating the cosine similarity between two vectors. The first step of LSA is to represent the text as a matrix. In this matrix, each row stands for a unique word and each column stands for a text passage or other context. Taking an example in Landauer et al. [17], if we have the data in Figure 2.12, we can compute the matrix as shown in Figure 2.13. Then, we use singular value decomposition (SVD) to decompose an $m \times n$ matrix M into a factorization of the form (2.27). In this equation, U is an $m \times m$ orthogonal matrix, V is an $n \times n$ orthogonal matrix, and Σ is an $m \times n$ diagonal matrix. SVD can filter out noisy elements and strengthen important values. Also, it reduces the dimensions of a matrix, making vector-space operation much more efficient. After applying SVD, each word is represented as a vector with n dimensions. For every word pair W_i and W_j, the distance between two term vectors W_i and W_j can be measured by cosine similarity (2.28). In this equation, w_{ik} and w_{jk} are the k-th components of vectors W_i and W_j, respectively.

$$M = U \times \Sigma \times V^T \tag{2.27}$$

$$cos(W_i, W_j) = \frac{W_i \cdot W_j}{|W_i||W_j|} = \frac{\sum_{k=1}^{n} w_{ik} w_{jk}}{\sqrt{\sum_{k=1}^{n} w_{ik}^2 \sum_{k=1}^{n} w_{jk}^2}} \tag{2.28}$$

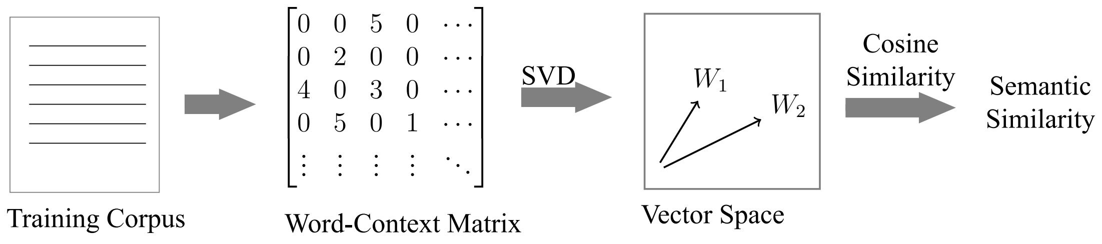
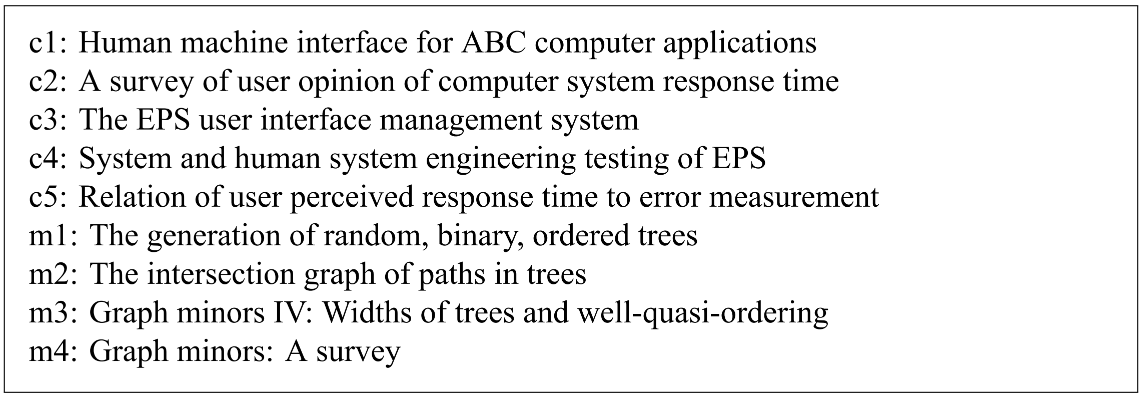


Figure 2.11: Procedure of Latent Semantic Analysis



c1: Human machine interface for ABC computer applications
c2: A survey of user opinion of computer system response time
c3: The EPS user interface management system
c4: System and human system engineering testing of EPS
c5: Relation of user perceived response time to error measurement
m1: The generation of random, binary, ordered trees
m2: The intersection graph of paths in trees
m3: Graph minors IV: Widths of trees and well-quasi-ordering
m4: Graph minors: A survey

Figure 2.12: Example of Text Data

	c1	c2	c3	c4	c5	m1	m2	m3	m4
human	1	0	0	1	0	0	0	0	0
interface	1	0	1	0	0	0	0	0	0
computer	1	1	0	0	0	0	0	0	0
user	0	1	1	0	1	0	0	0	0
system	0	1	1	2	0	0	0	0	0
response	0	1	0	0	1	0	0	0	0
time	0	1	0	0	1	0	0	0	0
EPS	0	0	1	1	0	0	0	0	0
survey	0	1	0	0	0	0	0	0	1
trees	0	0	0	0	0	1	1	1	0
graph	0	0	0	0	0	0	1	1	1
minors	0	0	0	0	0	0	0	1	1

Figure 2.13: Example of Terms-Contexts Matrix

Chapter 3

Related Works

In this chapter, we discuss some previous works related to our research. These works can be categorized into two types: works on computational semantics and works on RTE. Among the works of computational semantics, most of them are about English, and only few of them are about Chinese. Among the works of RTE, there are three adopted approaches. These approaches include formal-logic approach, machine learning approach and mixed approach. However, in the works of Chinese RTE, they only adopt machine learning approach. We will introduce these works in the following sections.

3.1 Computational Semantics

Research on computational semantics spans in several dimensions, such as semantic representation, semantic construction, inference techniques, knowledge resources and wide-coverage text understanding [19]. Logic form semantic representation is proposed by Montague [20], and is usually adopted. There are three types of methods for semantic construction, including lambda calculus based algorithm[6], unification-based algorithm [21] and linear logic [22]. There are two types of inference techniques, including theorem proving [23], and model building [24]. Lexical knowledge resources, such as WordNet [12], provide the semantic relations between words. The semantic relations between the synonym sets in WordNet can be transformed into logical forms. Extended-WordNet is a logic-form translation of WordNet [25]. Other lexical knowledge resources, such as

FrameNet [26], VerbNet [13] and PropBank [27], provide the annotations of thematic roles for Neo-Davidson semantics. Wide-coverage is an important issue for computational semantics when applied to real-word applications. Clark and Curra [28] construct semantic representations from the derivations produced by a statistical parser for combinatory categorical grammar (CCG) [29]. Crouch and King [30] construct semantic representations from functional structures obtained from parsing newswire text by lexical functional grammar (LFG) [31]. Copestake and Flickinger [32] construct semantic representations by a grammar development environment using head-driven phrase structure grammar (HPSG) [33].

3.1.1 Computational Semantics in Chinese

Most of the studies for computational semantics are about English, while only few of them are about Chinese.

Zhang and Zhang [34] present a logical approach toward answer validation in Chinese question answering system. They also introduce a transformation algorithm that transforms Chinese sentence into logical form. Their algorithm uses *translation codes* to build semantic representations from syntax tree. The translation codes are designed for a small set of grammar rules that contains about 370 rules that cover the entire set of parsing trees. This work also introduces a method to use HowNet [35], a Chinese lexical knowledge dictionary and taxonomy, as a substitution for WordNet, to provide lexical knowledge required for logical reasoning.

Chen and Wu [36] demonstrate a method to construct logical form from data in Academia Sinica Balanced Corpus [37]. They adopt a rule-based strategy to build the semantic grammar. The semantic grammar is constructed as two parts: a syntactic part that generates syntactic structures, and a semantic part that builds semantic representations. The semantic representation in this work includes the lambda notation and the first-order predicate logic with a Davidsonian analysis of event semantics and the quantifier semantics of Generalized Quantifier Theory [38].

3.2 Recognizing Textual Entailment

The task of RTE was first proposed in the FRACAS project [39]. The data in the FRACAS test suite is created artificially, and is designed for evaluating the quality of semantic representations by formal logic. RTE became popular when Dagan et al. [7] started a series of competitions of RTE task, known as the PASCAL RTE challenges. The data in PASCAL RTE is real-world data rather than artificially constructed examples. It is not designed for evaluating the quality of semantic representation, but for evaluating the performance of RTE task. Androutsopoulos and Malakasiotis [40] survey a wide variety of techniques for paraphrasing and RTE tasks used on the MSR (Microsoft Research Paraphrase Corpus). These techniques include logic-based approach, machine learning, vector space models (VSM), syntactic similarity, symbolic meaning representation similarity, and decoding. Mainly, these techniques can be categorized into three types: formal logic approach, machine learning approach and mixed approach.

3.2.1 Formal Logic Approach

Tatu and Moldovan [8] demonstrate a formal logic approach to RTE task. In this work, the first step is syntactic parsing. In order to reduce the error of parsing, the syntax trees are derived from two types of parsers: constituency parser and dependency parser, and then they construct the semantic representation from the two syntax trees. Lexical knowledge is derived from eXtended WordNet, and world knowledge is hand-coded built according to the development set data. This system achieves high accuracy on the 2006 PASCAL RTE data.

Wotzlaw and Coote [41] present a new system for recognizing textual entailment. This system is a modular environment allowing for a high-coverage syntactic and semantic text analysis combined with logical inference. For the syntactic and semantic analysis, they combine a deep semantic analysis with a shallow one supported by statistical models. The semantic representation in this work is minimal recursion semantics (MRS)[21], which is a common semantic formalism for HPSG grammar[33]. For RTE, they use logical inference of first-order logic employing model-theoretic techniques and automated reasoning tools. The external knowledge resources are provided for the reasoning procedure, including lexical semantics, such as WordNet[12], and ontology, such as YAGO[42] and OpenCyc[43].

3.2.2 Machine Learning Approach

Zanzotto et al. [44] introduce the class of pair feature spaces, which allows machine learning algorithms to exploit the relations between a text (T) and a hypothesis (H), and derive first-order rewrite rules on a syntactic-semantic representation of training examples. These features include lexical, syntactic and semantic features. They also propose a method to encode shallow semantic information in data representation through the use of typed anchors. Extensive experiments on the RTE challenge datasets: RTE1, RTE2 and RTE3, demonstrate that the proposed method works.

Huang et al. [45] introduce a method to model what human think in the RTE process to solve the RTE problem. First, they analyze a labeled RTE test set and find five significant negative entailment phenomena that are important features for RTE. Then, they propose a method to extract these kinds of phenomena from raw text automatically, and train the SVM classifiers with these features to solve the RTE problem. They evaluate the performance on both the English RTE-5 dataset and the Chinese NTCIR-9 RITE dataset, and conclude the same findings.

3.2.3 Mixed Approach

Bos and Markert [46] use logical inference techniques for recognizing textual entailment. They use a robust wide-coverage CCG-parser[28] to generate semantic representations for each T/H-pair. They also incorporate theorem proving techniques with model building techniques to solve RTE by formal logic method, and show that it is an useful and robust method for the approximate entailment. In addition, they use machine learning techniques to combine the formal semantic techniques with simple shallow word overlap.

The resulting hybrid model achieves high accuracy on the PASCAL RTE test sets.

Raina et al. [47] present a system for textual inference that uses learning and a logical-formula semantic representation of the text. First, they use dependency parser[48] to build dependency graph from raw data, and translate the relations represented in the dependency graph into a logical formula. Then, they use an abductive inference algorithm[49] to perform semantic reasoning. Also, they use machine learning techniques to learn how to combine information from several knowledge resources to decide which are plausible assumptions during its reasoning process. In the end, they do an experiment on the dataset from the PASCAL RTE 2005 challenge competition, and achieve the highest confidence weighted score.

3.2.4 Recognizing Textual Entailment in Chinese

Chinese RTE is a new open research issue. There are only a few prior works related to Chinese RTE, due to the lack of datasets and benchmarks. The RITE task in NTCIR-9 [50] and NTCIR-10 [51] provide a benchmark for researchers to evaluate the methods on this topic. All of the prior works in Chinese RIT use machine learning techniques.

Huang et al. [52] propose a system to deal with the Chinese RTE problem for NTCIR-9 RITE task. This system is based on the support vector machine (SVM) model, and they construct the feature vectors by finding the differences between the text T1 and text T2 on several linguistic levels, including lexical, syntactic, dependency and sentiment polarity difference.

Yang et al. [53] introduce their finding on the entailment analysis of the NTCIR-10 RITE-2 dataset, and use their observation to improve their system. They define ten special cases that are not suitable for conventional machine learning classifier. They also implement some modules for four special cases, and the result is significantly improved from 67.86 to 72.92 on the binary classification task.

Chapter 4

Implementation: System Architecture and Algorithm

In this chapter, we introduce the implementation of our system. This system consists of several components, including the CKIP Chinese Parser, Semantic Constructor, RTE Engine and knowledge resources. Also, we introduce the algorithm in each component. The CKIP Chinese Parser converts Chinese raw sentence into syntax tree. The Semantic Constructor constructs the semantic representation from syntax tree. The RTE Engine solves RTE problem by proving the satisfiability of input logical form. The knowledge resources provide the required knowledge for RTE Engine.

4.1 Overview

The System Architecture is shown in Figure 4.1. The input is raw text. It consists of the text pairs of T and H of a given RTE problem. Then, we use the CKIP Chinese Parser to segment and parse the raw text, and construct the syntax trees. In the next step, the Semantic Constructor builds the logical forms of T and H from the syntax trees, and then the RTE Engine solves the RTE problem by proving whether T entails H is true or false. In some cases, extra knowledge is required to complete the proof. The RTE Engine will construct the logical forms of required knowledge, and use other knowledge resources to validate them. In this system, there are two kinds of knowledge resources: lexical semantics and distributional semantics. Lexical semantics is a kind of handcraft resource for lexical knowledge. It is possible that the knowledge cannot be found in these resources. Then, we need to obtain the knowledge from the corpus by distributional semantics. When the knowledge is prepared for the RTE Engine, we can prove whether T entails H. We will elaborate on each component in Figure 4.1 in the following sections.

4.2 CKIP Chinese Parser

In order to construct the logical form of a Chinese sentence, we need to parse the sentence into a syntax tree first. The algorithm for parsing Chinese sentence is statistical parsing, which is a type of machine learning technique. By learning the parsing rules from large amount of human-annotated corpus, this algorithm can generate the most probable syntax tree of the input sentence. In this research, we use the CKIP Chinese Parser [54] to generate the syntax tree from the raw sentence. We take the sentence (4.1) as an example. For this example, the syntax tree generated by the CKIP Parser is shown in Figure 4.2.

小明玩電腦
XiaoMing plays computer. (4.1)

In this figure, the annotations of nodes can be categorized into two types: *label* and *tag*. There are two types of labels: *Head* and *Semantic Labels*. Head denotes the *Head Word* for a given tree or subtree. Semantic Label is the label that represents thematic roles, such as *agent* and *goal*. In this sentence, agent means subject, and goal means object. The tags *NP*, *VP*, etc. are *Part of Speech Tags* (POS tags) of words or subtrees. The detailed introduction of these semantic labels and POS tags are in Technique Report of Sinica CKIP Group [55, 56].

Also, the input of the CKIP Chinese Parser can be the segmented Chinese sentence, as in (4.2), to generate syntax tree in Figure 4.2. Since the segmentation result from the CKIP Parser may not be correct, we can reduce the segmentation error by taking the manually

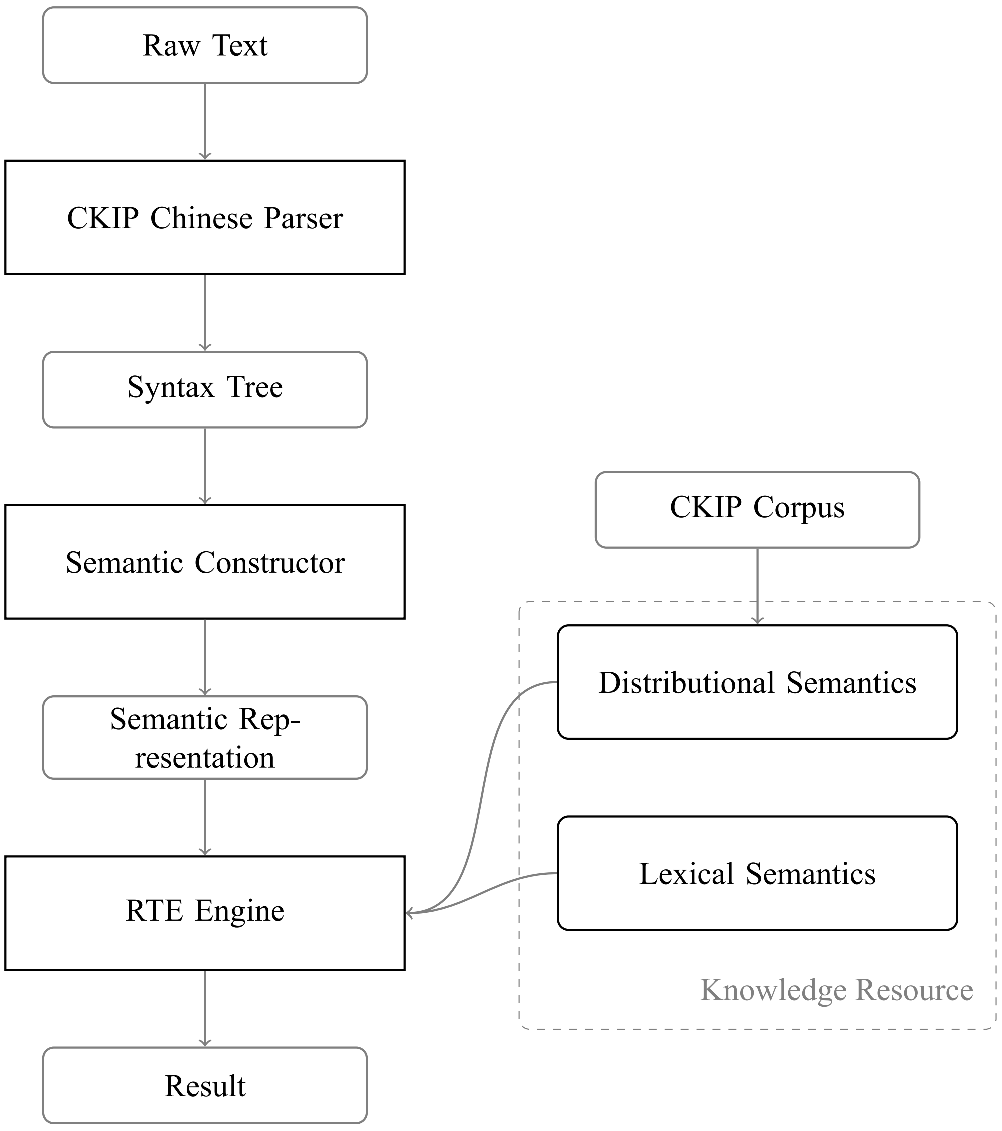


Figure 4.1: Overview of System Architecture

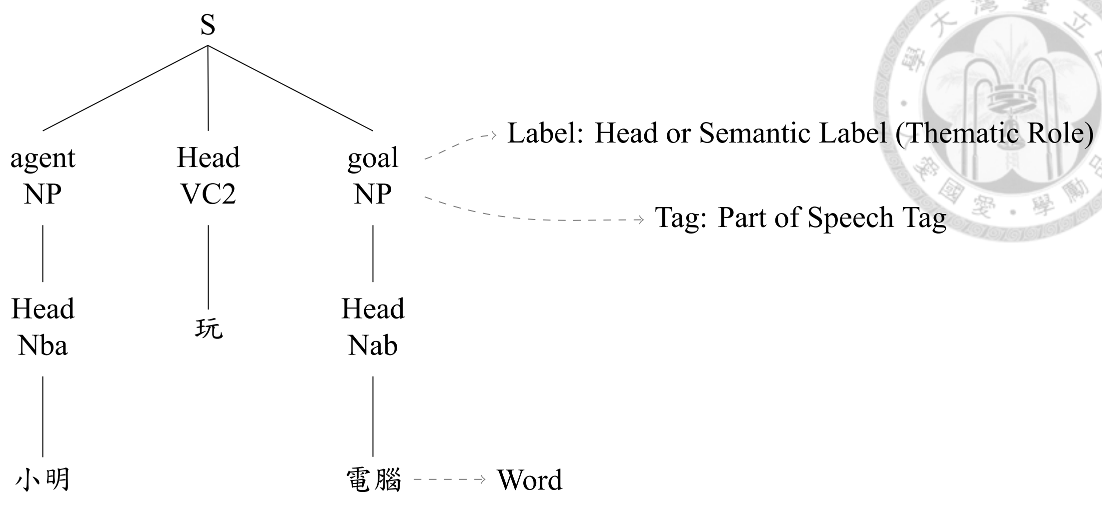


Figure 4.2: Syntax Tree from the CKIP Chinese Parser

segmented text as input for the CKIP Parser. This issue will be discussed in Section 5.1.

小明	玩	電腦
Xiaomin	Play	Computer

(4.2)

4.3 Semantic Constructor

The role of the Semantic Constructor is to convert the syntax tree from the CKIP Parser into logical form. We adopt the logical form of Neo-Davidsonian Semantics introduced in Section 2.2.2. We use lambda calculus to construct logical form from syntax tree, and integrate the annotations of semantic labels into logical form. The logical form constructed from the syntax tree in Figure 4.2 is shown in (4.3).

$$\text{小明}(n_1) \wedge agent(n_1, e) \wedge \text{玩}(e) \wedge \text{電腦}(n_2) \wedge goal(n_2, e) \tag{4.3}$$

We elaborate on how to construct this logical form in the following sections.

4.3.1 Semantic Construction

We present an algorithm to convert the syntax tree produced by the CKIP Chinese Parser into Neo-Davidsonian (introduced in Section 2.2.2) logical form by lambda calculus. This algorithm consists of five types of *operations*. These operations are listed in the following:

Op1 : Initialization

Op2 : Pass Operation

Op3 : Application with Semantic Role

Op4 : Conjunction of Multiple Children

Op5 : Termination

The overall process of this algorithm is depicted in Figure 4.3. In this figure, the dashed-line arrows represent the steps of semantic construction, and the dotted-line arrows represent the paths of tree traversal from leaves. We will elaborate on these operations in the following paragraphs.

Op1 : Initialization

The first operation is initialization. In this operation, we give semantic representations to every leaf with word and every node whose label is thematic role. The semantics of node with word W is $\lambda x.W(x)$, and the semantics of node with thematic role R is $\lambda P \lambda x \lambda e.R(x,e) \wedge P(x)$. For example, in Figure 4.3, the semantics of leaf node with word 小明 (XiaoMing) is $\lambda x.\text{小明}(x)$, and the semantics of node with role $agent$ is $\lambda P \lambda x \lambda e.agent(x,e) \wedge P(x)$. After initialization, we traverse this syntax tree up from every leaf.

Op2 : Pass Operation

After operation 1, we traverse up from every leaf. In this operation, if the label of the node is *Head* or *DUMMY*, and the node has only one child, we just need to pass the semantics from the child of this node to this node. In Figure 4.3, the parent node of the node with word 小明 has only one child, and its label is Head; hence, we just pass the semantics $\lambda x.\text{小明}(x)$ upward to this node.

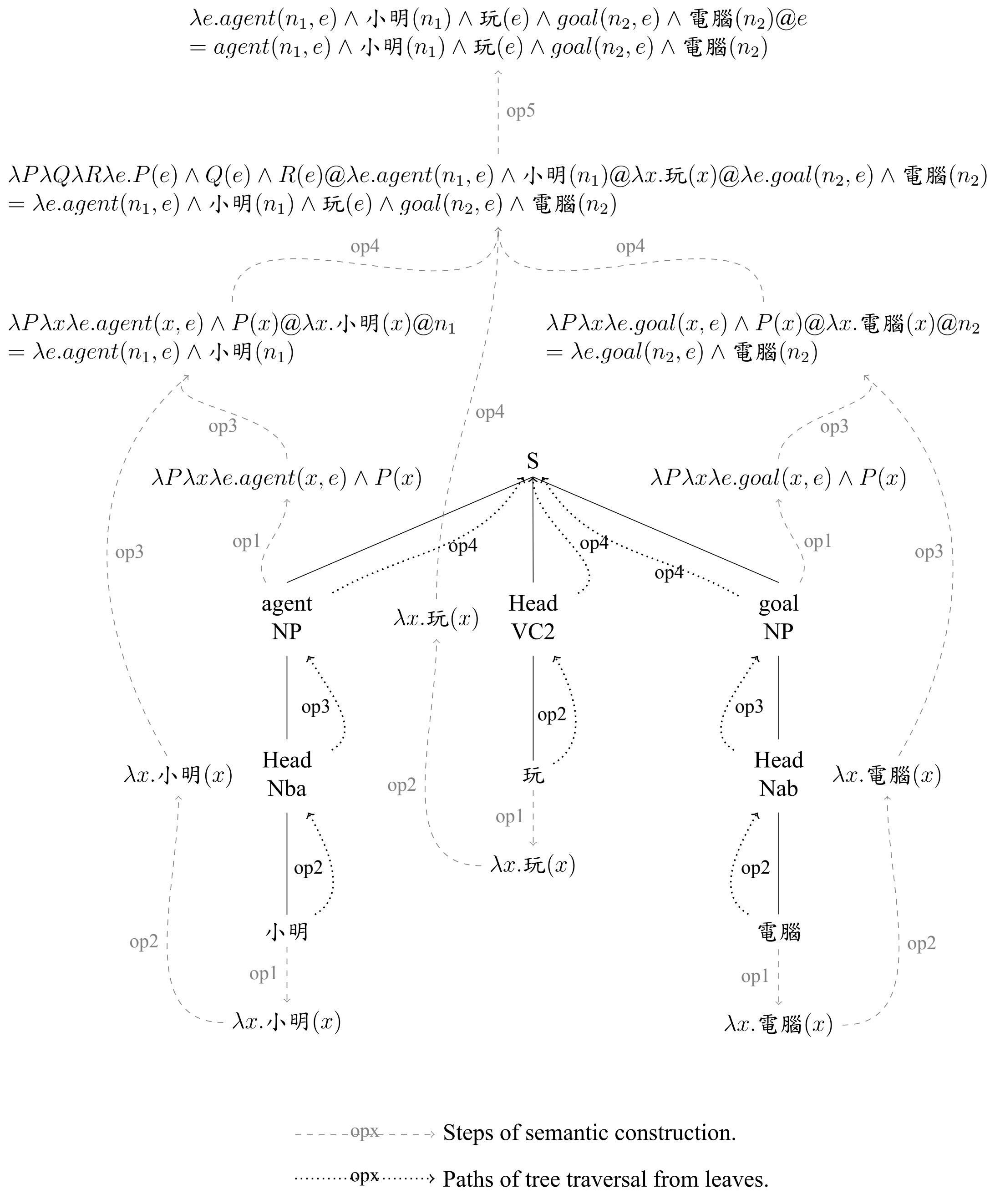

$\lambda e.agent(n_1,e) \wedge$ 小明$(n_1) \wedge$ 玩$(e) \wedge goal(n_2,e) \wedge$ 電腦$(n_2)@e$
$= agent(n_1,e) \wedge$ 小明$(n_1) \wedge$ 玩$(e) \wedge goal(n_2,e) \wedge$ 電腦(n_2)
op5
$\lambda P \lambda Q \lambda R \lambda e.P(e) \wedge Q(e) \wedge R(e)@\lambda e.agent(n_1,e) \wedge$ 小明$(n_1)@\lambda x.$玩$(x)@\lambda e.goal(n_2,e) \wedge$ 電腦(n_2)
$= \lambda e.agent(n_1,e) \wedge$ 小明$(n_1) \wedge$ 玩$(e) \wedge goal(n_2,e) \wedge$ 電腦(n_2)
op4
op4
$\lambda P \lambda x \lambda e.agent(x,e) \wedge P(x)@\lambda x.$小明$(x)@n_1$
$= \lambda e.agent(n_1,e) \wedge$ 小明(n_1)
$\lambda P \lambda x \lambda e.goal(x,e) \wedge P(x)@\lambda x.$電腦$(x)@n_2$
$= \lambda e.goal(n_2,e) \wedge$ 電腦(n_2)
op3
op4
op3
S
$\lambda P \lambda x \lambda e.agent(x,e) \wedge P(x)$
$\lambda P \lambda x \lambda e.goal(x,e) \wedge P(x)$
op3
op1
op4
op4
op4
op1
op3
agent NP
$\lambda x.$玩(x)
Head VC2
goal NP
op3
op2
op3
$\lambda x.$小明(x)
Head Nba
op2
玩
Head Nab
$\lambda x.$電腦(x)
op1
op2
$\lambda x.$玩(x)
op2
小明
電腦
op2
op1
op1
op2
$\lambda x.$小明(x)
$\lambda x.$電腦(x)
opx Steps of semantic construction.
opx Paths of tree traversal from leaves.

Figure 4.3: Algorithm for Semantic Construction

Op3 : Application with Semantic Role

While traversing the tree upward, if the label of the node is thematic role, and it has only one child, we combine the semantics of thematic role with the semantics of its child by application of lambda calculus. In Figure 4.3, the node with role $agent$ has semantics $\lambda P \lambda x \lambda e.agent(x,e) \wedge P(x)$, and the semantics of its child is $\lambda x.\text{小明}(x)$. The application of lambda calculus is shown in (4.4).

$$\lambda P \lambda x \lambda e.agent(x,e) \wedge P(x) @ \lambda x.\text{小明}(x) = \lambda x \lambda e.agent(x,e) \wedge \text{小明}(x) \tag{4.4}$$

After combining the semantics, we need to rename the variable x. For example, if the word 小明 is the first noun appearing in the sentences of RTE paris T and H, we need to replace the variable x of predicate 小明 with n_1, as shown in (4.5).

$$\lambda x \lambda e.agent(x,e) \wedge \text{小明}(x) @ n_1 = \lambda e.agent(n_1,e) \wedge \text{小明}(n_1) \tag{4.5}$$

Op4 : Conjunction of Multiple Children

When traversing the tree upward, if a node has multiple children, we need to combine the semantics of every child by conjunction. In Figure 4.3, the node with tag S has three children, and we use application of lambda calculus with the formula $\lambda P \lambda Q \lambda R \lambda e.P(e) \wedge Q(e) \wedge R(e)$ to combine the semantics of child nodes, as shown in (4.6).

$$\begin{aligned} &\lambda P \lambda Q \lambda R \lambda e.P(e) \wedge Q(e) \wedge R(e) \\ &@ \lambda e.agent(n_1,e) \wedge \text{小明}(n_1) @ \lambda x.\text{玩}(x) @ \lambda e.goal(n_2,e) \wedge \text{電腦}(n_2) \\ &= \lambda e.agent(n_1,e) \wedge \text{小明}(n_1) \wedge \text{玩}(e) \wedge goal(n_2,e) \wedge \text{電腦}(n_2) \end{aligned} \tag{4.6}$$

Op5 : Termination

When traversing the tree upward, if we reach the root of the tree, we need to do the operation of termination. This operation is to remove the remaining λ operator in logical form. For example, in Figure 4.3, the node S is the root of the tree, so we remove the remaining λ operator by application, as shown in (4.7).

$$\begin{aligned}&\lambda e.agent(n_1, e) \wedge \text{小明}(n_1) \wedge \text{玩}(e) \wedge goal(n_2, e) \wedge \text{電腦}(n_2)@e \\ &= agent(n_1, e) \wedge \text{小明}(n_1) \wedge \text{玩}(e) \wedge goal(n_2, e) \wedge \text{電腦}(n_2)\end{aligned} \tag{4.7}$$

4.3.2 Semantic Construction for Prepositional Phrase

In some cases, the sentence may contain Prepositional Phrase (PP). For example, the sentence in (4.8) contains a PP: 在 (at) 家 (home),

小明 (XiaoMing) 在 (at) 家 (home) 玩 (play) 電腦 (computer) (4.8)

and the syntax tree of this sentence is in Figure 4.4. If we use the previous algorithm to

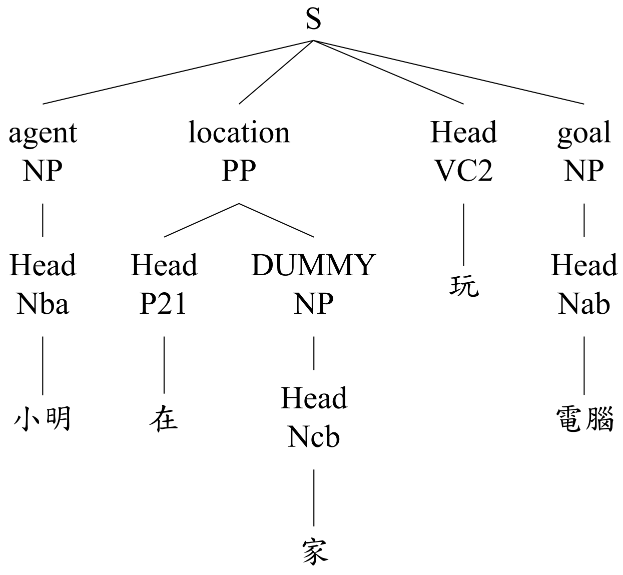


Figure 4.4: Syntax Tree of Sentence with Prepositional Phrase

generate logical form from Figure 4.4, we will get a redundant atomic formula 在(n_1), as shown in (4.9).

$$\begin{aligned}&\text{小明}(n_0) \wedge agent(n_0, e) \wedge \text{在}(n_1) \wedge \text{家}(n_1) \\ &\wedge location(n_1, e) \wedge \text{玩}(e) \wedge \text{電腦}(n_2) \wedge goal(n_2, e)\end{aligned} \tag{4.9}$$

Since the relation between 家(n_1) and 玩(e) is annotated by the thematic role $location(n_1, e)$, the formula 在(n_1) becomes redundant. In order to avoid creating the redundant formula, we need to check the tags during the operation Op1. We should check the tag of the leaf before initializing its semantics. If the first alphabet of the tag is *P*, it means that this word is a preposition. Hence, we should not initialize the semantic of this leaf. We illustrate this process in Figure 4.5. The first step is Op1. In this step, we should not need to ini-

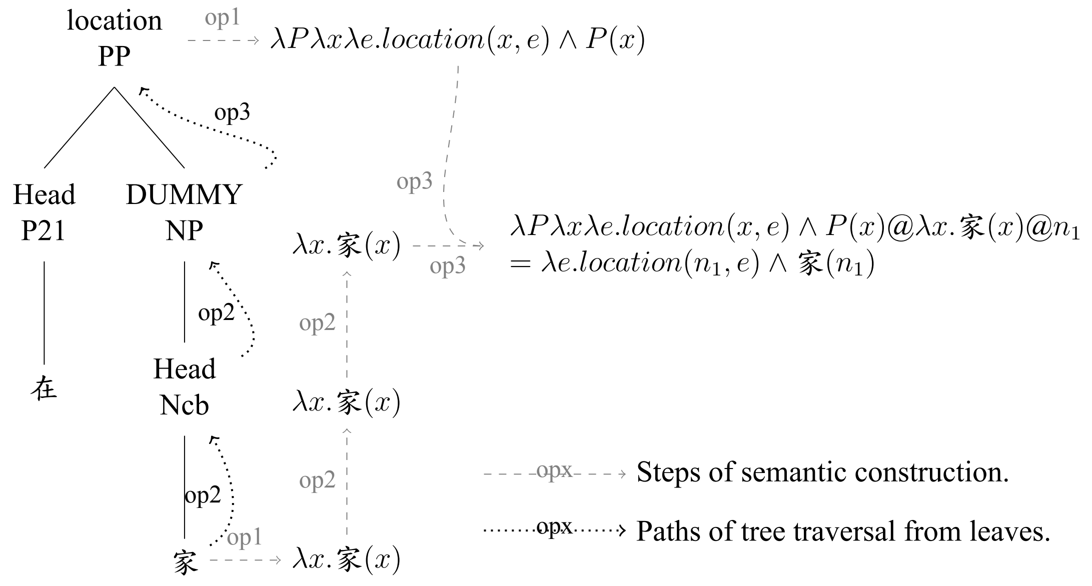

location
PP
op1
$\lambda P \lambda x \lambda e.location(x, e) \wedge P(x)$
op3
Head
P21
DUMMY
NP
在
op2
Head
Ncb
op2
op1
家
$\lambda x.家(x)$
op2
$\lambda x.家(x)$
op2
$\lambda x.家(x)$
op3
op3
$\lambda P \lambda x \lambda e.location(x, e) \wedge P(x) @ \lambda x.家(x) @ n_1$
$= \lambda e.location(n_1, e) \wedge 家(n_1)$
opx
Steps of semantic construction.
opx
Paths of tree traversal from leaves.

Figure 4.5: Semantic Construction for Prepositional Phrase

tialize the semantics of the leaf with word 在, because the tag of this word is *P21*. We only need to initialize the semantic of word 家 as $\lambda x.家(x)$ and initialize the semantic of thematic role *location* as $\lambda P \lambda x \lambda e.location(x, e) \wedge P(x)$. The next step is to traverse up the tree from its leaves. Since there is only one leaf 家 with semantic representation, we only need to traverse up from it. In addition, the labels of parent and grandparent of this node are HEAD and DUMMY, respectively. They satisfy the requirement of Op2, so we only need to pass the semantic representation up to its grandparents. The last step, when we reach the root of the tree, since this role only has a child with semantic representation, we only need to do Op3. By using Op3, we combine the semantics of thematic role with the semantics of its child, and generate the final result, as shown in (4.10).

$$\lambda e.location(n_1, e) \wedge 家(n_1) \tag{4.10}$$

Comparing to the logical form in (4.9), this result does not contain the redundant formula $在(n_1)$. It can be further combined with the semantic representations of other nodes in the syntax tree Figure 4.4, and generate the final result (4.11).

$$\text{小明}(n_0) \wedge agent(n_0, e) \wedge \text{家}(n_1) \wedge location(n_1, e) \\ \wedge \text{玩}(e) \wedge \text{電腦}(n_2) \wedge goal(n_2, e) \tag{4.11}$$

4.3.3 Semantic Construction for Noun Phrase

In some cases, the noun phrase (NP) is composed by multiple words. For example, in (4.12), the NP, 電腦 (computer) 遊戲 (game), consists of two words.

小明 (XiaoMing) 玩 (play) 電腦 (computer) 遊戲 (game) (4.12)

We build the syntax tree of this sentence, as illustrated in Figure 4.6. To construct the semantic representation of NP 電腦 (computer) 遊戲 (game), we can use the same algorithm introduced in Section 4.3.1. The process of generating the logical form from the subtree of this NP is shown in Figure 4.7. The first step is Op1. In this step, we initialize the se-

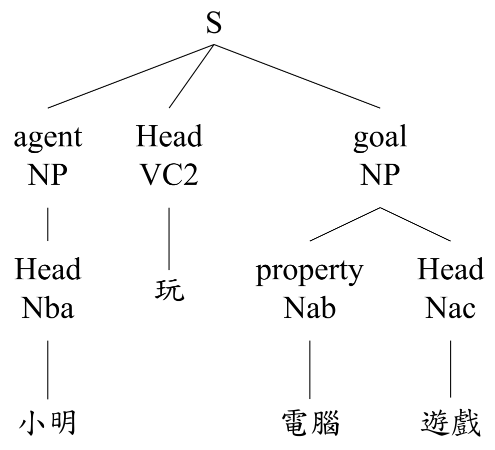


Figure 4.6: Syntax Tree of Sentence with Noun Phrase

mantic representations of two words: 電腦 and 遊戲, and two labels: *goal* and *property*. The next step is to traverse up the tree from its leaves. In the left branch with word 電腦, the parent has label *property*, so we use Op3 to combine their logical forms, as shown in (4.13).

$$\lambda P \lambda x \lambda e.property(x, e) \wedge P(x) @ \lambda x.\text{電腦}(x) @ n_1 \\ = \lambda e.property(n_1, e) \wedge \text{電腦}(n_1) \tag{4.13}$$

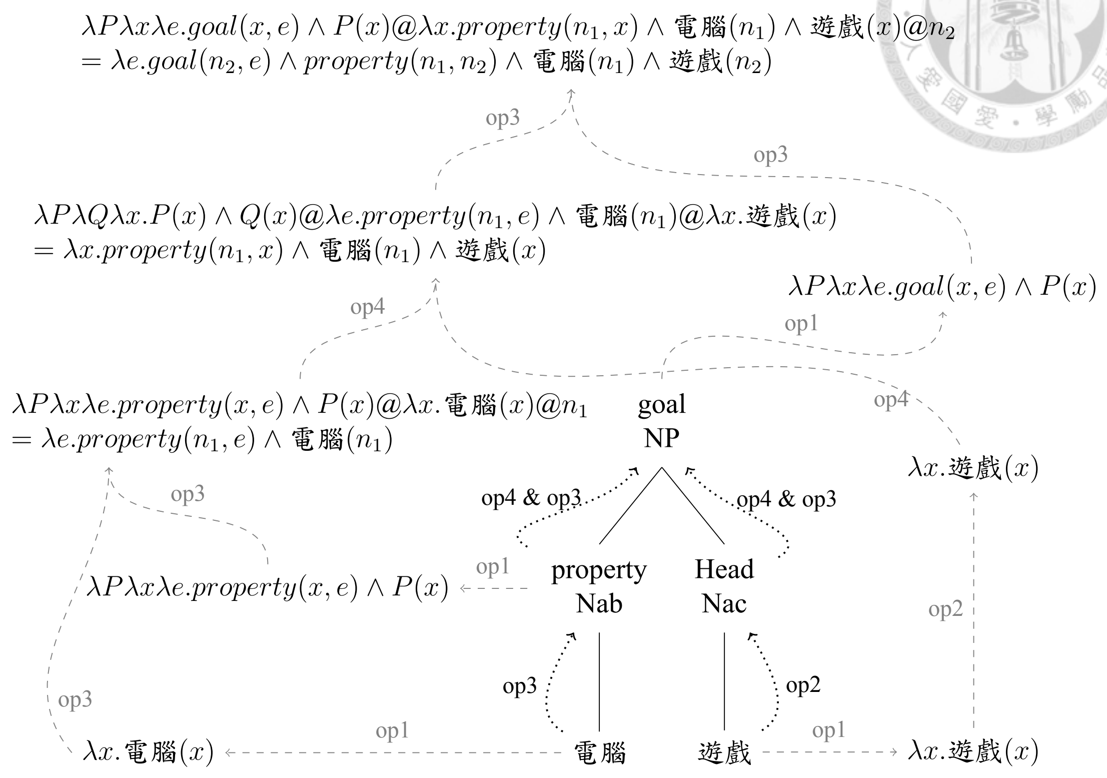

$\lambda P\lambda x\lambda e.goal(x,e) \wedge P(x)@\lambda x.property(n_1,x) \wedge$ 電腦$(n_1) \wedge$ 遊戲$(x)@n_2$
$= \lambda e.goal(n_2,e) \wedge property(n_1,n_2) \wedge$ 電腦$(n_1) \wedge$ 遊戲(n_2)
op3
op3
$\lambda P\lambda Q\lambda x.P(x) \wedge Q(x)@\lambda e.property(n_1,e) \wedge$ 電腦$(n_1)@\lambda x.$遊戲(x)
$= \lambda x.property(n_1,x) \wedge$ 電腦$(n_1) \wedge$ 遊戲(x)
$\lambda P\lambda x\lambda e.goal(x,e) \wedge P(x)$
op4
op1
$\lambda P\lambda x\lambda e.property(x,e) \wedge P(x)@\lambda x.$電腦$(x)@n_1$
$= \lambda e.property(n_1,e) \wedge$ 電腦(n_1)
goal
NP
op4
$\lambda x.$遊戲(x)
op3
op4 & op3
op4 & op3
$\lambda P\lambda x\lambda e.property(x,e) \wedge P(x)$
op1
property
Nab
Head
Nac
op2
op3
op2
op3
op1
op1
$\lambda x.$電腦(x)
電腦
遊戲
$\lambda x.$遊戲(x)

Figure 4.7: Semantic Construction for Noun Phrase

In the right branch with word 遊戲, the label of the parent is *HEAD*, so we use Op2 to pass the semantic representation to the parent. In the next step, we traverse up to the root. Since the root has two children, and they all have their own semantic representations. We need to use Op4 to combine their semantic representations first. We use the logical form $\lambda P\lambda Q\lambda x.P(x) \wedge Q(x)$ to combine the semantics in two branches, shown in (4.14).

$$\begin{aligned} &\lambda P\lambda Q\lambda x.P(x) \wedge Q(x)@\lambda e.property(n_1,e) \wedge \text{電腦}(n_1)@\lambda x.\text{遊戲}(x) \\ &= \lambda x.property(n_1,x) \wedge \text{電腦}(n_1) \wedge \text{遊戲}(x) \end{aligned} \tag{4.14}$$

After we combine the semantics in these two branches, we have not finished this procedure yet. Since the root has the thematic role *goal*, we need to use Op3 to combine the result of (4.14) with the logical form of the thematic role at root. This operation is shown in (4.15), and it produces the final result of logical form of this NP.

$$\begin{aligned}&\lambda P \lambda x \lambda e.goal(x,e) \wedge P(x) @ \lambda x.property(n_1,x) \wedge 電腦(n_1) \wedge 遊戲(x) @ n_2 \\ &= \lambda e.goal(n_2,e) \wedge property(n_1,n_2) \wedge 電腦(n_1) \wedge 遊戲(n_2)\end{aligned} \tag{4.15}$$

With this logical form, we can combine it with other nodes in the syntax tree in Figure 4.6, and generate the final result (4.16).

$$\begin{aligned}&小明(n_0) \wedge agent(n_0,e) \wedge 玩(e) \\ &\wedge goal(n_2,e) \wedge property(n_1,n_2) \wedge 電腦(n_1) \wedge 遊戲(n_2)\end{aligned} \tag{4.16}$$

Notice that the formula $property(n_1, n_2)$ is not the so-called Neo-Davidsonian semantic representation, because the argument n_2 does not represent an event.

4.3.4 Semantic Construction for DE Phrase

In Chinese, the word 的 (DE), is a functional word that links a noun to an entity to represent the owner of the entity, or links a noun or adjective to an entity to represent the property of the entity. For example, in (4.17a), the DE word represents that the computer is the father's computer, and in (4.17b), the DE word represents that the computer is the latest computer.

爸爸 (father) 的 (DE) 電腦 (computer) (4.17a)

最新 (latest) 的 (DE) 電腦 (computer) (4.17b)

When a DE word exists in a sentence, we need to notice that we should not create its semantic representation as a redundant atomic formula. For example, the sentence (4.18) has the word 的.

小明 (XiaoMing) 玩 (play) 爸爸 (father) 的 (DE) 電腦 (computer) (4.18)

The syntax tree of this sentence is shown in Figure 4.8. In order to build the logical form without redundant atomic formula of the DE word, we can treat the DE word as the

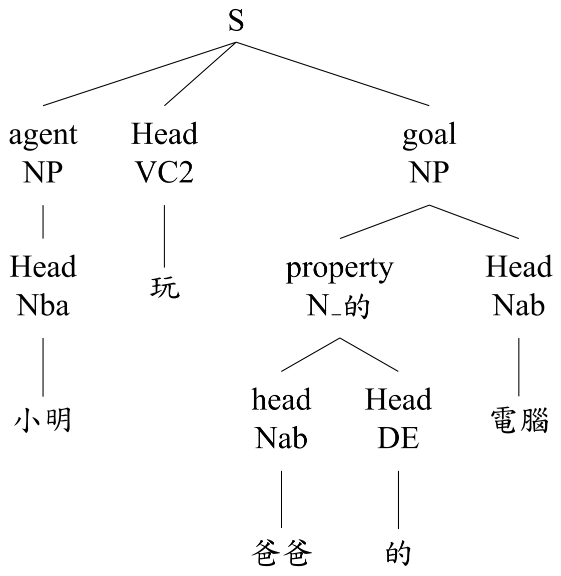


Figure 4.8: Syntax Tree of Sentence with DE

preposition mentioned in Section 4.3.2. By adopting the same method in Section 4.3.2, we can build the logical form of the DE phrase: 爸爸的電腦 (father's computer), as shown in (4.19).

$$\lambda e.goal(n_2, e) \wedge \text{爸爸}(n_0) \wedge property(n_0, n_2) \wedge \text{電腦}(n_2) \tag{4.19}$$

Then, with this logical form, we can generate the logical form of syntax tree in Figure 4.8, and the final result is shown in (4.20).

$$\begin{aligned} &\text{小明}(n_1) \wedge agent(n_1, e) \wedge \text{玩}(e) \\ &\wedge \text{爸爸}(n_0) \wedge property(n_0, n_2) \wedge \text{電腦}(n_2) \wedge goal(n_2, e) \end{aligned} \tag{4.20}$$

4.3.5 Semantic Construction for Embedded Sentence

In some cases, a sentence can be embedded within another sentence. This phenomenon is called embedded sentence. For example, the sentence (4.21) contains an embedded sentence: 小明玩電腦 (XiaoMing plays computer).

小華 (XiaoHwa) 看 (see) 小明 (XiaoMing) 玩 (play) 電腦 (computer) (4.21)

The CKIP Chinese Parser can generate the syntax tree of this example, as illustrated in Figure 4.9, and the tag of the embedded sentence is *S*. We can also build the logical form of

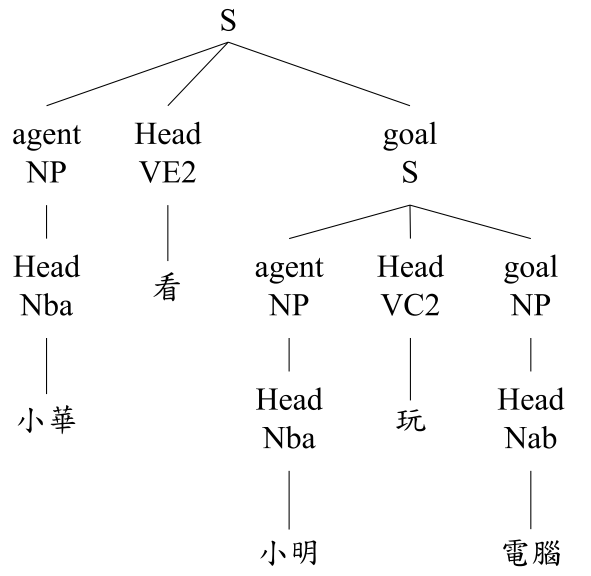


Figure 4.9: Syntax Tree of Sentence with Embedded Sentence

this syntax tree by the following procedure. First, we build the semantic representation of the embedded sentence, and this process can be done if we use the algorithm in Figure 4.3. However, in this case, we should not use the operation Op5 to terminate the procedure. The resulting logical form is shown in (4.22). In the next step, we use the procedure shown in Figure 4.10.

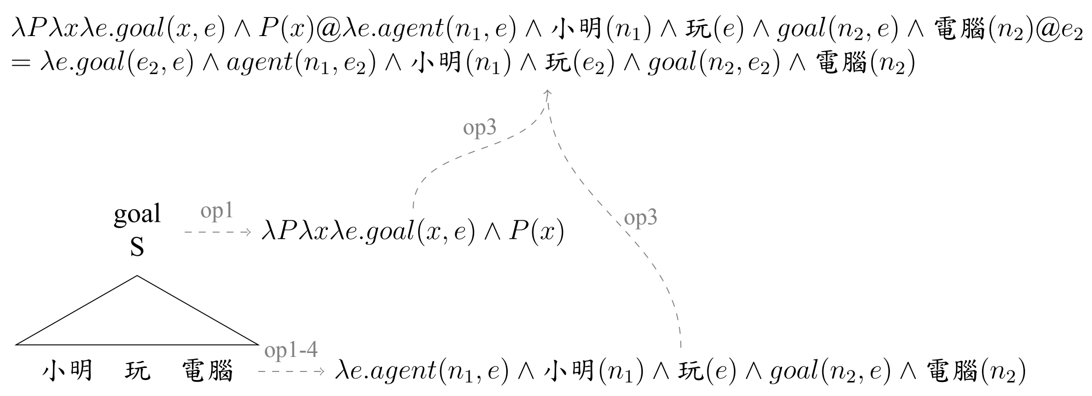
$$\lambda e.agent(n_1, e) \wedge \text{小明}(n_1) \wedge \text{玩}(e) \wedge goal(n_2, e) \wedge \text{電腦}(n_2) \tag{4.22}$$

At the beginning, we initialize the semantics of the role *goal* by Op1, and then we use Op3 to combine the semantics of the role and the embedded sentence, as shown in (4.23).

$$\begin{aligned}
&\lambda P \lambda x \lambda e.goal(x, e) \wedge P(x) \\
&@\lambda e.agent(n_1, e) \wedge \text{小明}(n_1) \wedge \text{玩}(e) \wedge goal(n_2, e) \wedge \text{電腦}(n_2)@e_2 \\
&\; = \lambda e.goal(e_2, e) \wedge agent(n_1, e_2) \wedge \text{小明}(n_1) \wedge \text{玩}(e_2) \wedge goal(n_2, e_2) \wedge \text{電腦}(n_2)
\end{aligned} \tag{4.23}$$

Notice that we replace the variable e in (4.22) by the variable e_2 during the operation of Op3, since the verb in the embedded sentence is not the same as the verb in its parent. After constructing the logical form in (4.23), we can combine it with its parent sentence in Figure 4.9, and generate the final result (4.24).

$$\begin{aligned}&\text{小華}(n_0) \wedge agent(n_0, e_1) \wedge \text{看}(e_1) \wedge \text{小明}(n_1) \wedge agent(n_1, e_2)\\ &\wedge \text{玩}(e_2) \wedge \text{電腦}(n_2) \wedge goal(n_2, e_2) \wedge goal(e_2, e_1)\end{aligned} \tag{4.24}$$



$\lambda P \lambda x \lambda e.goal(x, e) \wedge P(x) @ \lambda e.agent(n_1, e) \wedge \text{小明}(n_1) \wedge \text{玩}(e) \wedge goal(n_2, e) \wedge \text{電腦}(n_2) @ e_2$
$= \lambda e.goal(e_2, e) \wedge agent(n_1, e_2) \wedge \text{小明}(n_1) \wedge \text{玩}(e_2) \wedge goal(n_2, e_2) \wedge \text{電腦}(n_2)$
op3
op3
goal
S
op1
$\lambda P \lambda x \lambda e.goal(x, e) \wedge P(x)$
小明 玩 電腦
op1-4
$\lambda e.agent(n_1, e) \wedge \text{小明}(n_1) \wedge \text{玩}(e) \wedge goal(n_2, e) \wedge \text{電腦}(n_2)$

Figure 4.10: Semantic Construction for Embedded Sentence

4.3.6 Semantic Construction for Negation

If a negation word exists in a sentence, the semantic representation of this sentence should be negated. For example, the sentence in (4.25) contains a negation word 沒有 (not).

$$\text{小明 (XiaoMing) 沒有 (not) 玩 (play)} \tag{4.25}$$

The syntax tree in Figure 4.11 shows that the thematic role of the negation word 沒有 is $negation$. We can build the semantics of this sentence by the procedure demonstrated in Figure 4.12. In the first step Op1, we should not initialize the semantics of the negation word. Then, we traverse up from the leaves and use Op2 and Op3 to pass and combine the semantics in other nodes. When we reach the root of tree, before we use Op4 to combine the semantics, we check the number of children containing the role of $negation$. If the number is odd, we add the sign of negation to the semantic template in Op4, as shown in (4.26), otherwise, we do not add the sign of negation.

$$\lambda P \lambda Q \lambda e.\neg(P(e) \wedge Q(e)). \tag{4.26}$$

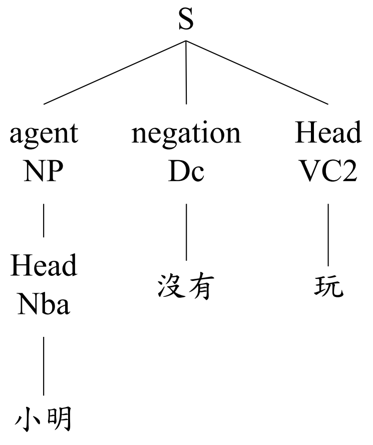


Figure 4.11: Syntax Tree of Sentence with Negation Word

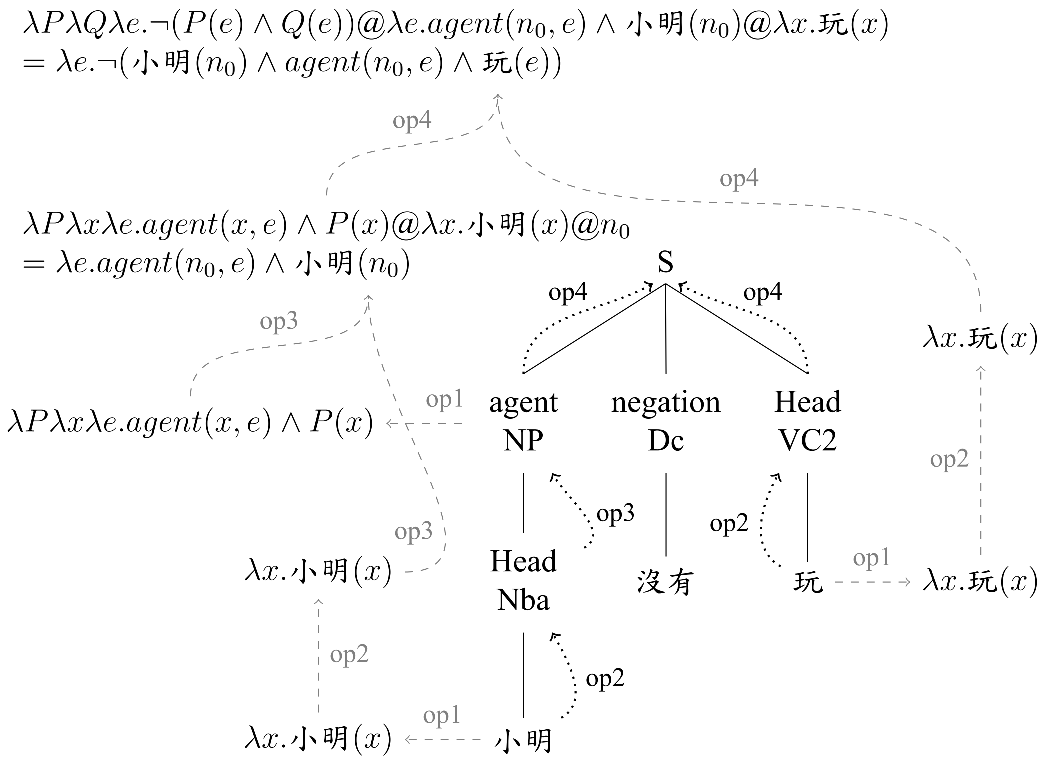

λPλQλe.¬(P(e) ∧ Q(e))@λe.agent(n0, e) ∧ 小明(n0)@λx.玩(x)
= λe.¬(小明(n0) ∧ agent(n0, e) ∧ 玩(e))
op4
op4
λPλxλe.agent(x, e) ∧ P(x)@λx.小明(x)@n0
= λe.agent(n0, e) ∧ 小明(n0)
S
op4
op4
λx.玩(x)
op3
λPλxλe.agent(x, e) ∧ P(x)
op1
agent
NP
negation
Dc
Head
VC2
op2
op3
op3
op2
λx.小明(x)
Head
Nba
沒有
玩
op1
λx.玩(x)
op2
op2
λx.小明(x)
op1
小明

Figure 4.12: Semantic Construction for Negation

Then, we use this template to combine the semantics from the child nodes, as shown in (4.27).

$$\begin{aligned}&\lambda P \lambda Q \lambda e.\neg(P(e) \wedge Q(e))@\lambda e.agent(n_0, e) \wedge 小明(n_0)@\lambda x.玩(x)\\&= \lambda e.\neg(小明(n_0) \wedge agent(n_0, e) \wedge 玩(e))\end{aligned} \tag{4.27}$$

To complete this procedure, we use Op5 in the last step, and the resulting semantic representation is shown in (4.28).

$$\neg(\text{小明}(n_0) \wedge agent(n_0, e) \wedge \text{玩}(e)) \tag{4.28}$$

Double Negation

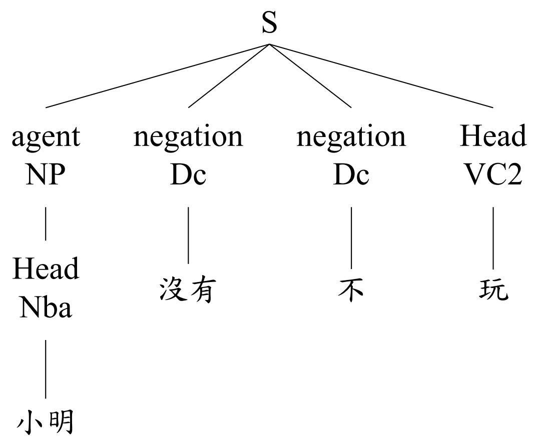


Figure 4.13: Syntax Tree of Sentence with Double Negation Words

The reason why we check the number of children containing the role of $negation$ is that it is possible to have more than one negation word in a sentence. For example, in (4.29), this sentence contains two negation words: 沒有 (not) and 不 (not).

小明 (XiaoMing) 沒有 (not) 不 (not) 玩 (play) (4.29)

This situation results in two children with negation roles in the syntax tree, as illustrated in Figure 4.13. Since the node S contains two child nodes with negation roles, in Op4, we do not need to add the sign of negation to the template. As a result, the resulting semantic representation will not contain the sign of negation, as shown in (4.30).

$$\text{小明}(n_0) \wedge agent(n_0, e) \wedge \text{玩}(e) \tag{4.30}$$

4.3.7 Semantic Construction for Coordination

If a sentence contains a coordinating word, we should use logical operator to represent the meaning of the coordinating word. For example, in (4.31), this sentence contains the coordinating word 或 (or).

小明 (XiaoMing) 玩 (play) 電腦 (computer) 或 (or) 手機 (cell-phone) (4.31)

The syntax tree of this example is shown in Figure 4.14. We build the semantic repre-

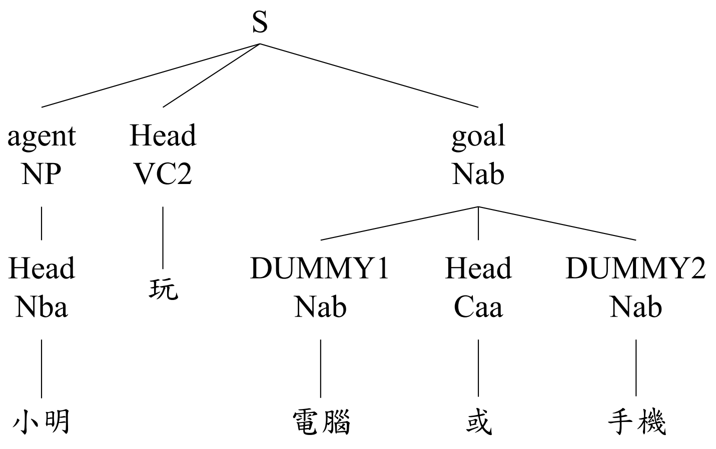


Figure 4.14: Syntax Tree of Sentence with Coordinating Word

sentation of coordinating subtree by the procedure shown in Figure 4.15. The first step is Op1, but we should not initialize the semantics of coordinating word in this step. Then we traverse up from other leaves and adopt Op2 to pass the semantics. When we reach the root of this subtree, we check the tag of child whose role is $HEAD$. If the tag is Caa, it means that the head word is a coordinating word. Then, we check whether the meaning of the word means conjunction or disjunction. In this case, the meaning of word 或 (or) means disjunction. Before we use Op4 to combine the semantics from both children, we need to use Op3 on both children respectively, as shown in (4.32).

$$\begin{aligned}&\lambda P \lambda x \lambda e.goal(x,e) \wedge P(x)@\lambda x.\text{電腦}(x)@(n_1)\\&= \lambda e.goal(n_1,e) \wedge \text{電腦}(n_1)\end{aligned} \tag{4.32a}$$

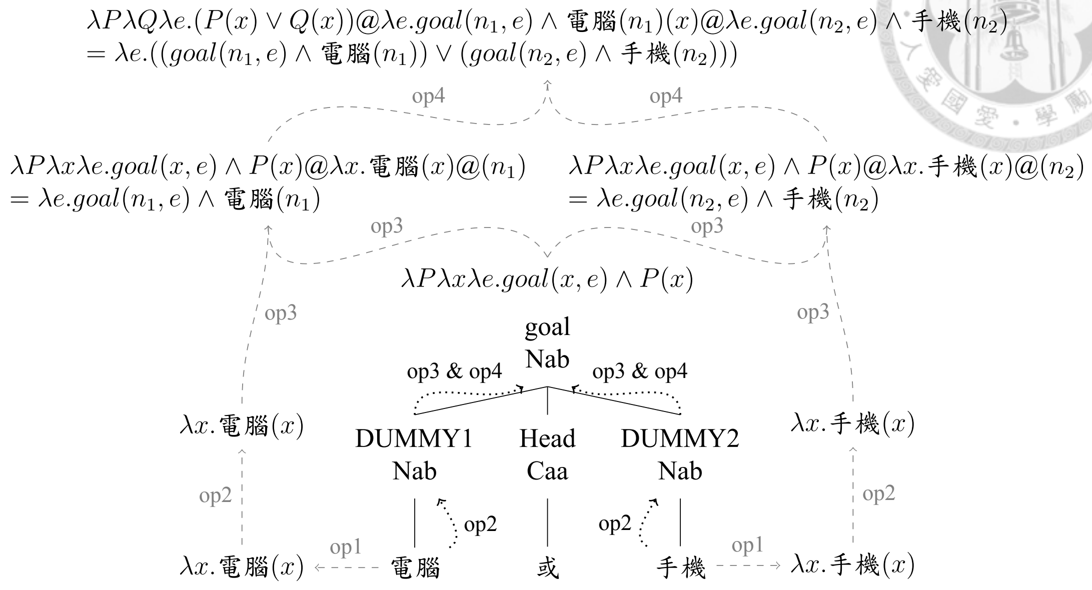

$\lambda P\lambda Q\lambda e.(P(x) \vee Q(x))@\lambda e.goal(n_1,e) \wedge \text{電腦}(n_1)(x)@\lambda e.goal(n_2,e) \wedge \text{手機}(n_2)$
$= \lambda e.((goal(n_1,e) \wedge \text{電腦}(n_1)) \vee (goal(n_2,e) \wedge \text{手機}(n_2)))$
op4
op4
$\lambda P\lambda x\lambda e.goal(x,e) \wedge P(x)@\lambda x.\text{電腦}(x)@(n_1)$
$= \lambda e.goal(n_1,e) \wedge \text{電腦}(n_1)$
$\lambda P\lambda x\lambda e.goal(x,e) \wedge P(x)@\lambda x.\text{手機}(x)@(n_2)$
$= \lambda e.goal(n_2,e) \wedge \text{手機}(n_2)$
op3
op3
$\lambda P\lambda x\lambda e.goal(x,e) \wedge P(x)$
op3
op3
goal
Nab
op3 & op4
op3 & op4
$\lambda x.\text{電腦}(x)$
DUMMY1
Nab
Head
Caa
DUMMY2
Nab
$\lambda x.\text{手機}(x)$
op2
op2
op2
op2
op1
op1
$\lambda x.\text{電腦}(x)$
電腦
或
手機
$\lambda x.\text{手機}(x)$

Figure 4.15: Semantic Construction for Coordination

$$\begin{aligned}&\lambda P\lambda x\lambda e.goal(x,e) \wedge P(x)@\lambda x.\text{手機}(x)@(n_2)\\&= \lambda e.goal(n_2,e) \wedge \text{手機}(n_2)\end{aligned} \tag{4.32b}$$

After this operation, we use the template (4.33) that contains disjunction in Op4 to combine the semantics between the two child nodes.

$$\lambda P\lambda Q\lambda x.(P(x) \vee Q(x)) \tag{4.33}$$

The operation to combine semantics in Op4 is in (4.34).

$$\begin{aligned}&\lambda P\lambda Q\lambda e.(P(x) \vee Q(x))@\lambda e.goal(n_1,e) \wedge \text{電腦}(n_1)(x)@\lambda e.goal(n_2,e) \wedge \text{手機}(n_2)\\&= \lambda e.((goal(n_1,e) \wedge \text{電腦}(n_1)) \vee (goal(n_2,e) \wedge \text{手機}(n_2)))\end{aligned} \tag{4.34}$$

In the end, we can use the result of (4.34) to build the semantic representation of the example in (4.31). The final result is in (4.35).

$$\text{小明}(n_0) \wedge agent(n_0,e) \wedge \text{玩}(e) \wedge ((\text{電腦}(n_1) \wedge goal(n_1,e)) \vee (\text{手機}(n_2) \wedge goal(n_2,e))) \tag{4.35}$$

4.3.8 Concluding Example

At the end of this section, we demonstrate the capability of our algorithm by constructing the logical form of an example. This example contains all the circumstances mentioned above. This example is shown in (4.36).

小華 (XiaoHwa) 跟 (to) 同學 (classmate) 說 (say) 隔壁班 (next class) 的 (DE)

小明 (XiaoMing) 不 (not) 玩 (play) 電腦 (computer) 和 (and) 手機 (cellphone) (4.36)

The syntax tree of this example is shown in Figure 4.16. The logical form constructed

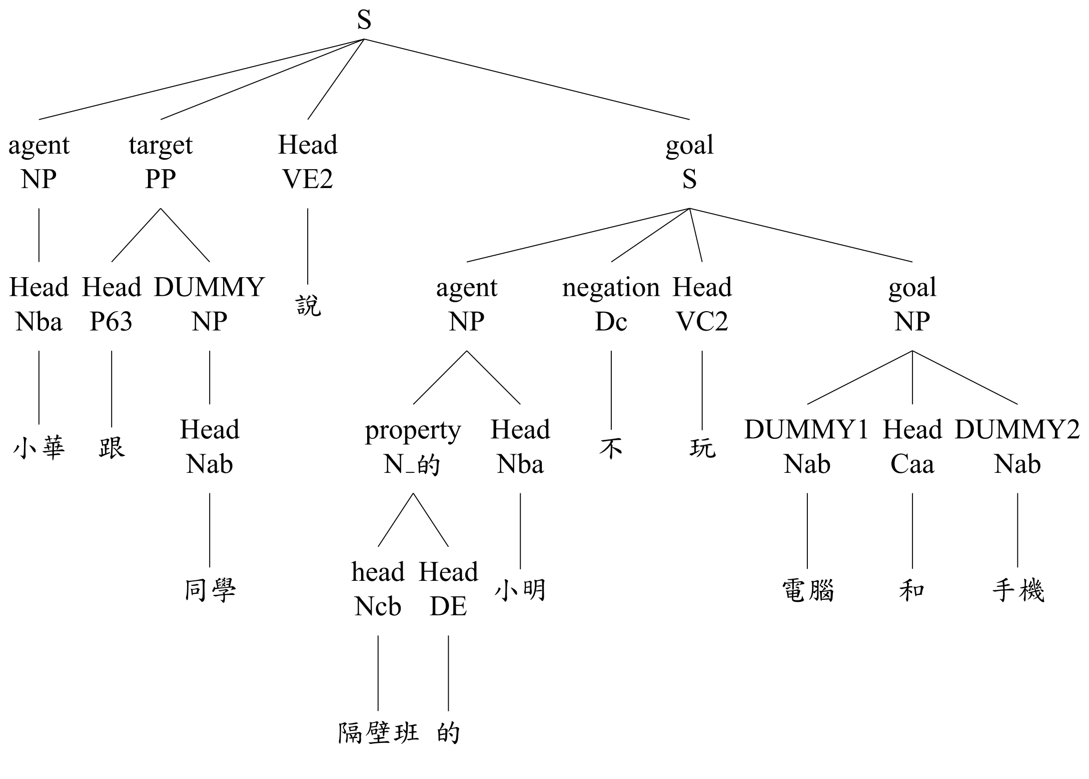


Figure 4.16: Syntax Tree of Sentence in Concluding Example

from this syntax tree is in (4.37).

$$\text{小華}(n_5) \wedge agent(n_5, e) \wedge \text{同學}(n_6) \wedge target(n_6, e) \wedge \text{說}(e) \wedge \neg(\text{隔壁班}(n_0) \wedge property(n_0, n_1) \wedge \text{小明}(n_1) \wedge agent(n_1, e_2) \wedge \text{玩}(e_2) \wedge \text{電腦}(n_3) \wedge goal(n_3, e_2) \wedge \text{手機}(n_4) \wedge goal(n_4, e_2)) \wedge goal(e_2, e) \quad (4.37)$$

4.4 Knowledge Resources

Before we introduce the RTE Engine, we need to introduce the knowledge resources first. There are two types of knowledge resources in our implementation. They are lexical semantics and distributional semantics. In this work, we do not adopt ontology as the knowledge resource, and the reason is discussed in Section 5.3.2.

4.4.1 Lexical Semantics

We use the Chinese WordNet [57] as Chinese lexical knowledge resource. Chinese WordNet is a Chinese version of WordNet. In Chinese WordNet, each word is linked to one or more lemmas, and each lemma has one or more senses. For example, the word 狗 (gou3) is linked to two lemmas, 狗^1, and 狗^2. The lemma 狗^1 has nine senses, including *canis familiaris*, *one of the Chinese Animal Zodiac*, *someone who is despicable*, etc. The sense *canis familiaris* has a synonym 犬 (quan3). These semantic relations are shown in Figure 4.17.

4.4.2 Distributional Semantics

We use the Chinese Latent Semantic Analysis (Chinese LSA) Website [58] as the knowledge source of distributional semantics. It calculates the value of LSA from the Sinica Corpus [37]. There are several parameters available to be adjusted, such as comparison type: Term-to-Term or Term-to-Document, dimensions of vector space, Language: Traditional Chinese or Simplified Chinese, etc. For example, if we want to compare the semantic similarity between two pairs of words: 老師 (teacher), 教師 (instructor) and 老師 (teacher), 醫師 (doctor). We set the comparison type as Term-to-Term, dimension as 300, and Language as Traditional Chinese. The result is shown in Table 4.1, and the semantic similarity between teacher and instructor is stronger than the semantics similarity between doctor and teacher, as expected. Hence, it is possible to use the Chinese LSA Website to check whether a pair of words is similar to each other or not.

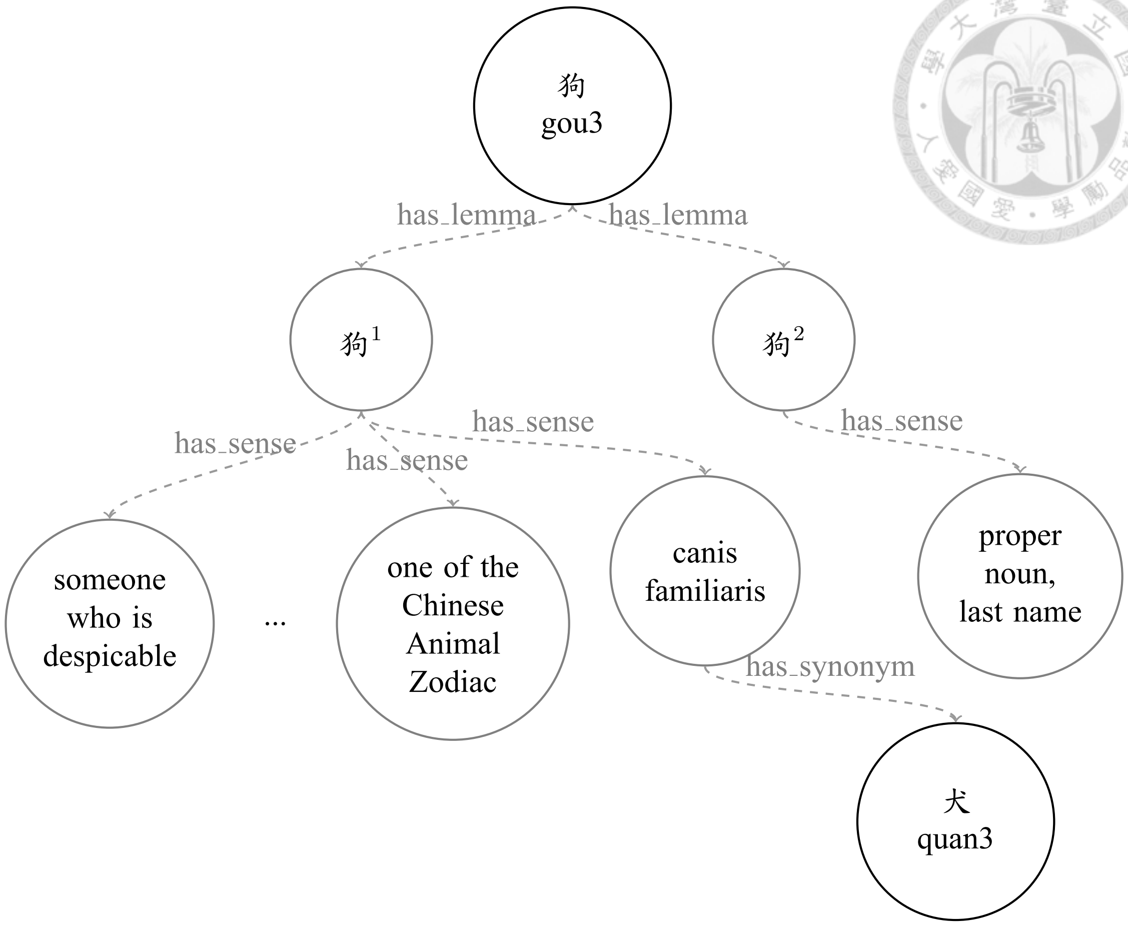


Figure 4.17: Example of Semantic Relations in Chinese WordNet

Pair of Words	Cosine Similarity
老師 (teacher), 教師 (instructor)	0.291650605182
老師 (teacher), 醫師 (doctor)	0.0233063926297

Table 4.1: Example of Semantic Similarity

4.5 RTE Engine

In order to solve the RTE problem, first, we need to convert both T and H into logical forms. The RTE Engine takes the logical forms of T and H of the RTE problem as input, and returns the result of whether T entails H. For example, if we want to solve the RTE problem in (4.38), we need to convert it into logical forms (4.39).

T 小明 (XiaoMing) 在 (at) 家 (home) 玩 (play) 電腦 (computer)
H 小明 (XiaoMing) 玩 (play) 電腦 (computer) (4.38)

$$\begin{aligned} T \quad & \text{小明}(n_0) \wedge agent(n_0, e) \wedge \text{家}(n_2) \wedge location(n_2, e) \\ & \wedge \text{玩}(e) \wedge \text{電腦}(n_1) \wedge goal(n_1, e) \\ H \quad & \text{小明}(n_0) \wedge agent(n_0, e) \wedge \text{玩}(e) \wedge \text{電腦}(n_1) \wedge goal(n_1, e) \end{aligned} \tag{4.39}$$

In some cases, extra knowledge is required to solve the entailment of T to H. For example, in (4.40), the logical form of extra knowledge K (4.41), is required to solve whether T entails H.

T	sentence	小明 (XiaoMing) 在 (at) 家 (home) 玩 (play) 電玩 (computer game)
	logical form	$\text{小明}(n_0) \wedge agent(n_0, e) \wedge \text{家}(n_2) \wedge location(n_2, e)$
		$\wedge \text{玩}(e) \wedge \text{電玩}(n_1) \wedge goal(n_1, e)$
H	sentence	小明 (XiaoMing) 玩 (play) 電動 (computer)
	logical form	$\text{小明}(n_0) \wedge agent(n_0, e) \wedge \text{玩}(e) \wedge \text{電動}(n_3) \wedge goal(n_3, e)$

(4.40)

$$K : \text{電玩}(n_1) \wedge goal(n_1, e) \rightarrow \text{電動}(n_3) \wedge goal(n_3, e) \tag{4.41}$$

The RTE Engine, will automatically construct the logical form of required knowledge K. By using the knowledge resources, the correctness of K can be validated.

The architecture of the RTE Engine is illustrated in Figure 4.18. The first step is to check whether we need to add extra knowledge or not, and this process is done by the Knowledge Builder. If extra knowledge is not required, we use the theorem prover to prove whether $T \rightarrow H$ is true or false. Otherwise, if extra knowledge is required, the Knowledge Builder will build the logical form of required knowledge K. Then, we use the Knowledge Validator to check whether the required knowledge in logical form K is valid. If K is valid, we use the theorem prover to prove the truth of $T \wedge K \rightarrow H$. Otherwise, we drop the logical form K, and use the theorem prove the truth of $T \rightarrow H$. We will elaborate on this procedure in the following paragraphs.

4.5.1 Knowledge Builder

The task of the Knowledge Builder is to check whether it is required to add extra knowledge or not. If the extra knowledge is required, it builds the logical form of required

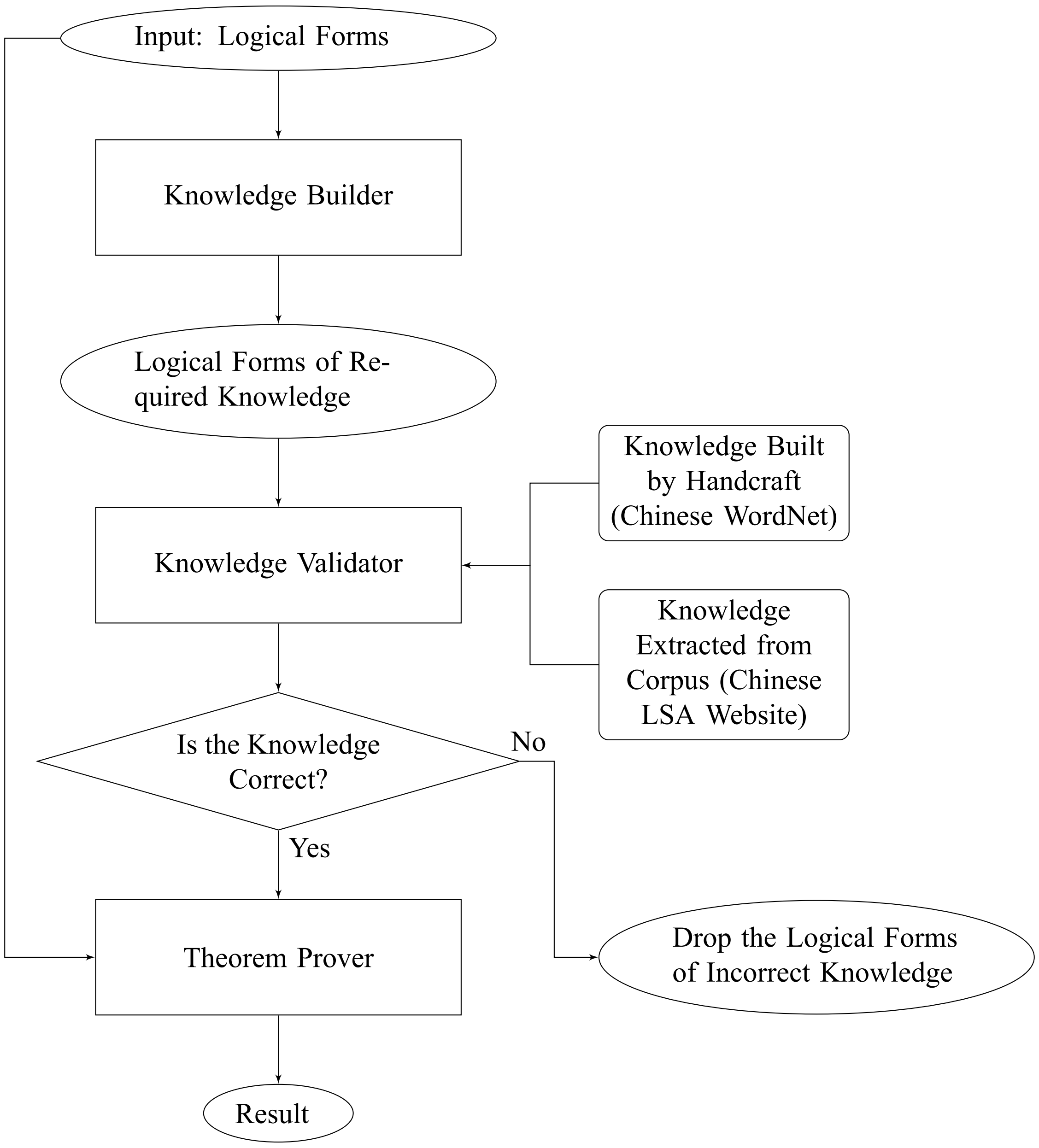

Input: Logical Forms
Knowledge Builder
Logical Forms of Re- quired Knowledge
Knowledge Validator
Knowledge Built by Handcraft (Chinese WordNet)
Knowledge Extracted from Corpus (Chinese LSA Website)
Is the Knowledge Correct?
No
Yes
Theorem Prover
Drop the Logical Forms of Incorrect Knowledge
Result

Figure 4.18: Architecture of RTE Engine

```
   input  : T,H
   output: K
1  K = Array();
2  for word w in T do
3      if w not in H then
4          if w is verb then
5              v = verb in H;
6              add w → v to K;
7          else if thematic role of w is in H then
8              r = thematic role of w;
9              v = word in H with thematic role r;
10             add w ∧ r → v ∧ r to K;
11         end
12     end
13 end
```

Figure 4.19: Algorithm for Knowledge Builder

knowledge. In this work, we use a very simple algorithm to check and build the logical form, and this algorithm is shown in Figure 4.19. In this algorithm, line 1 initializes the array K for storing the logical forms of required knowledge. In line 2-13, we use a **for** loop to check whether the word in w is in H or is not in H. The conditions related to this loop can be categorized into several cases, and the corresponding examples are in Table 4.2.

Case 1 In line 4, w satisfies the condition that w is not in H, we check whether w is a verb or not. If it is a verb, it belongs to this case. For example, in Table 4.2a, the verb 玩 (play) in T is not in H. In order to make that T entails H become true, we need to add a logical form of extra knowledge indicating that the verb in T entails the verb in H. The logical form of this extra knowledge, k_1, is in (4.42). It defines that the verb 玩 in T entails the verb 打 (play) in H, and it makes $T \wedge k_1 \rightarrow H$ become true. In this case, we need to add k_1 into array K.

$$k_1 : \text{玩}(e) \rightarrow \text{打}(e) \tag{4.42}$$

Case 2 In line 7, w satisfies the condition that w is not in H, and w is not a verb. We check whether the thematic role of w is in H or not. If the thematic role is in H, it belongs to this case. For example, in Table 4.2b, the word 電玩 (computer game) in T is not in H,

T	sentence	小明 (XiaoMing) 在 (at) 家 (home) 玩 (play) 電玩 (computer game)
	logical form	小明$(n_0) \wedge agent(n_0, e) \wedge$ 家$(n_2) \wedge location(n_2, e)$
		$\wedge$玩$(e) \wedge$ 電玩$(n_1) \wedge goal(n_1, e)$
H	sentence	小明 (XiaoMing) 打 (play) 電玩 (computer game)
	logical form	小明$(n_0) \wedge agent(n_0, e) \wedge$ 打$(e) \wedge$ 電玩$(n_1) \wedge goal(n_1, e)$

(a) Case 1

T	sentence	小明 (XiaoMing) 在 (at) 家 (home) 玩 (play) 電玩 (computer game)
	logical form	小明$(n_0) \wedge agent(n_0, e) \wedge$ 家$(n_2) \wedge location(n_2, e)$
		$\wedge$玩$(e) \wedge$ 電玩$(n_1) \wedge goal(n_1, e)$
H	sentence	小明 (XiaoMing) 玩 (play) 電動 (computer game)
	logical form	小明$(n_0) \wedge agent(n_0, e) \wedge$ 玩$(e) \wedge$ 電動$(n_3) \wedge goal(n_3, e)$

(b) Case 2

T	sentence	小明 (XiaoMing) 昨天 (yesterday) 玩 (play) 電腦 (computer)
	logical form	小明$(n_0) \wedge agent(n_0, e) \wedge$ 昨天$(n_3) \wedge time(n_3, e) \wedge$ 玩(e)
		$\wedge$電腦$(n_1) \wedge goal(n_1, e)$
H	sentence	小明 (XiaoMing) 在 (at) 家 (home) 玩 (play) 電腦 (computer)
	logical form	小明$(n_0) \wedge agent(n_0, e) \wedge$ 家$(n_2) \wedge location(n_2, e)$
		$\wedge$玩$(e) \wedge$ 電腦$(n_1) \wedge goal(n_1, e)$

(c) Case 3

T	sentence	小明 (XiaoMing) 在 (at) 家 (home) 玩 (play) 電玩 (computer game)
	logical form	小明$(n_0) \wedge agent(n_0, e) \wedge$ 家$(n_2) \wedge location(n_2, e)$
		$\wedge$玩$(e) \wedge$ 電玩$(n_1) \wedge goal(n_1, e)$
H	sentence	小明 (XiaoMing) 打 (play) 電動 (computer game)
	logical form	小明$(n_0) \wedge agent(n_0, e) \wedge$ 打$(e) \wedge$ 電動$(n_3) \wedge goal(n_3, e)$

(d) Case 4

T	sentence	小明 (XiaoMing) 玩 (play) 電腦 (computer)
	logical form	小明$(n_0) \wedge agent(n_0, e) \wedge$ 玩$(e) \wedge$ 電腦$(n_1) \wedge goal(n_1, e)$
H	sentence	小明 (XiaoMing) 在 (at) 家 (home) 玩 (play) 電腦 (computer)
	logical form	小明$(n_0) \wedge agent(n_0, e) \wedge$ 家$(n_2) \wedge location(n_2, e)$
		$\wedge$玩$(e) \wedge$ 電腦$(n_1) \wedge goal(n_1, e)$

(e) Case 5

Table 4.2: Example in Checking for Missing Knowledge

but its thematic roles $goal$ is in H. We can build the logical form of extra knowledge, k_2, to make $T \wedge k_2 \rightarrow H$ become true. The extra knowledge is shown in (4.43). It contains the word 電玩 in T, its thematic role $goal$, and the word 電動 (computer game) in H that has the same thematic role. In this case, we need to add k_2 into array K.

$$k_2: 電玩(n_1) \wedge goal(n_1, e) \rightarrow 電動(n_3) \wedge goal(n_3, e) \tag{4.43}$$

Case 3 If w is not in H, and does not satisfy the **if** and **else if** condition in line 4 and line 7, logical form of extra knowledge is not required to add to K. For example, in Table 4.2c, the word 昨天 (yesterday) in T is not in H, and its thematic role is not in H, too.

Case 4 After running the **for** loop in line 2-13, it is possible that we need to add more than one logical form of extra knowledge into the array K. For example, in Table 4.2d, we need to add both k_1 in (4.42) and k_2 in (4.43) into K, to make $T \wedge k_1 \wedge k_2 \rightarrow H$ become true.

Case 5 Also, it is possible that K is empty. If K is empty, we do not need to add any logical form of extra knowledge. For example, in Table 4.2e, every word in T are in H.

4.5.2 Knowledge Validator

After the logical forms of required knowledge are built by Knowledge Builder, since we do not know whether the knowledge of these logical forms is correct or not, we need to validate their correctness. In this system, the Knowledge Validator can validate the correctness of the knowledge. The algorithm for the Knowledge Validator is shown in Figure 4.20. The input of the Knowledge Validator is the array Kin, which is the array K produced by the Knowledge Builder, and the output is $Kout$, which stores the valid logical forms that can be delivered to the Theorem Prover to solve whether T entails H. In line 1, we initialize $Kout$ as an empty array. In line 2-12, we validate every logical form of knowledge k in array Kin. If k is valid, we add it into $Kout$. In line 3, we extract the pair of words from the knowledge k. For example, in the logical form of

```
   input  : Kin
   output: Kout
1  Kout = Array() ;
2  for k in Kin do
3      w1, w2 = words in k;
4      valid = False ;
5      if checkCwn(w1,w2) then
6          valid = True ;
7      else if checkCLSA(w1,w2) then
8          valid = True ;
9      end
10     if valid is True then
11         add k to Kout;
12     end
13 end
```

Figure 4.20: Algorithm for Knowledge Validator

knowledge (4.43), we extract two words, $w1$ = 電玩 and $w2$ = 電動. The word $w1$ is the word on the left hand side of symbol $\rightarrow$, and $w2$ is the word on the right hand side. In line 4-12, we check whether k is valid by two functions: $checkCwn$ and $checkCLSA$. These two functions take the two words $w1$ and $w2$ as input, and check the validity of the knowledge by checking the semantic relation between $w1$ and $w2$. The semantic relation can be obtained from the Chinese WordNet and the Chinese LSA Website, respectively. After checking the validity, these two functions will return true if it is valid; otherwise, return false. If k is valid, we add k to $Kout$. We will introduce these two functions in the following paragraphs.

checkCwn

This function checks the validity of knowledge by the Chinese WordNet (introduced in Section 4.4.1). In this function, we check whether $w1$ entails $w2$ by checking whether $w2$ is the *hypernym* or *synonym* of $w1$, or not. If one of the senses of $w1$ is the hypernym or synonym of $w2$, the function will return true to indicate the validity of this knowledge. Hence, if $w2$ is the hypernym or synonym of $w1$, this knowledge is valid. Otherwise, it is not valid. For example, in Figure 4.17, if $w1$ = 狗 and $w2$ = 犬, the word 狗 has three senses, and one of these senses has synonym 犬, and this function will return true to indicate the validity of this knowledge. However, since a word can have multiple senses, it is possible that we will choose a wrong sense and generate an incorrect result. We will discuss this issue in Section 5.3.2.

checkCLSA

For some cases, we can't find the semantic relation between $w1$ and $w2$ in the Chinese WordNet, so we need to check its validity from other resources. This function checks the validity from the Chinese LSA Website (introduced in Section 4.4.2). The value return from this website is not the semantic relation between $w1$ and $w2$, but a numerical value of semantic similarity. For example, given $w1$ = 電玩 and $w2$ = 電動, the semantic similarity between $w1$ and $w2$ is 0.606751983694. However, we do not know whether this numerical value is higher enough to indicate that $w1$ entails $w2$. We need to compare the numerical value with a predefined threshold. The value of the threshold is chosen by try-and-error. Given lots of pairs of hypernym and synonym words, we found that the desirable threshold is around 0.1. If the semantic similarity is larger than 0.1, it is possible that $w2$ is synonym or hypernym of $w1$, and this function will return true to indicate the validity of this knowledge.

4.5.3 Theorem Prover

As described earlier in Section 4.5, we prove whether T entails H by using the theorem prover to prove $T \rightarrow H$. If the array K is not empty, it means that extra knowledge is required to complete the proof. In this situation, we use the theorem prover to prove the logical form in (4.44).

$$T \wedge (\bigwedge_{k_i \in K} k_i) \rightarrow H. \tag{4.44}$$

In this logical form, we provide the logical forms of extra knowledge to theorem prover by making the conjunction of all the logical forms of extra knowledge k_i in K with T. The theorem prover adopted by this work is the theorem prover in the python nltk module [59]. In the python nltk module, two types of theorem prover are provided: TableauProver and

ResolutionProver. The algorithm in TableauProver is tableaux algorithm, introduced in Section 2.4.1, and the algorithm in ResolutionProver is resolution algorithm, introduced in Section 2.4.2. Also, the python nltk provides the interface to an external theorem prover: Prover9 [60]. Prover9 is a resolution/paramodulation automated theorem prover for first-order and equational logic. According to our experiment, both TableauProver and Prover9 can solve the problems in reasonable time, while ResolutionProver takes longer time to solve the problems. However, in this work, we only care about whether the problems can be solved successfully, and the runtime is not the main concern for the present work.

Chapter 5

Implementation Issues

In this chapter, we discuss some situations in which our implementation can't solve the RTE problem successfully. There are several situations that our system will failed to solve the RTE problem. First, the syntax tree from the CKIP Chinese Parser can be erroneous. Second, the Semantic Constructor may not be able to construct the semantic representation of some specific words. Third, given the correct logical form, the RTE Engine may not solve the RTE due to several reasons. In the following sections, we will discuss these situations in more detail.

5.1 CKIP Chinese Parser

The CKIP Parser produces a syntax tree with thematic role labeling, that enables the semantic construction of Neo-Davidsonian semantic representation. However, there are some limitations about this parser. These limitations place a restriction on applying our work to real-world application. We elaborate on these limitations in the following sections.

5.1.1 Error in Syntax Tree

It is possible that the syntax tree generated by the CKIP Chinese Parser is incorrect, especially when parsing long sentence in real-world text. We categorize these errors into two types: error in Chinese word segmentation, and error in syntax tree construction.

Error in Chinese Word Segmentation

Unlike English, when processing Chinese text, we need to segment it before parsing. Then, we can build the syntax tree from the result of segmentation. However, when the result of segmentation is wrong, the syntax tree build from this result will not be correct. In real-world text, it is likely to have wrong segmentation when proper noun exists in the sentence. For example, we randomly select a sentence from news, as shown in (5.1).

檢方又多告葉世文一條財產來源不明罪 (5.1)

The correct segmentation result of sentence (5.1) is shown in (5.2).

檢方｜又｜多｜告｜葉世文｜一條｜財產｜來源｜不明罪 (5.2)

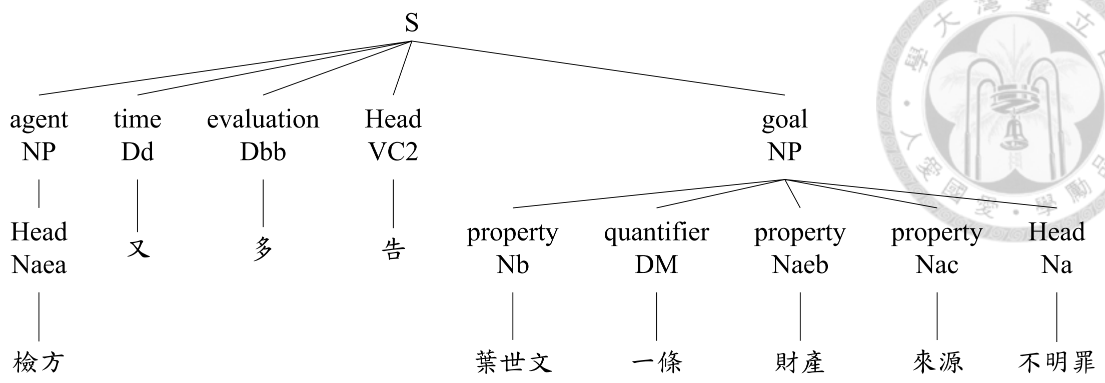
However, the segmentation result from CKIP Chinese Parser is not correct, as shown in (5.3).

檢方｜又｜多｜告葉世文｜一條｜財產｜來源｜不明罪 (5.3)

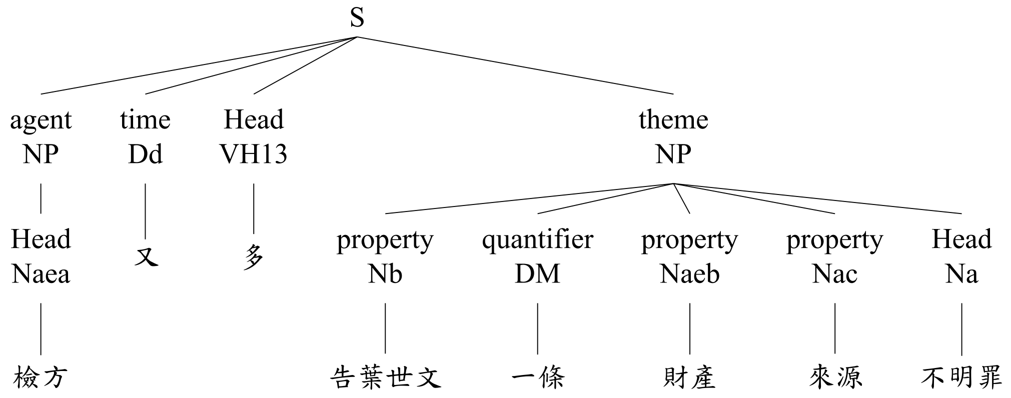
The word 告葉世文 (accuse Ye Shi-wen) in (5.3) should be split into two words: 告 (accuse) and 葉世文 (Ye Shi-wen), as shown in (5.2). Since 葉世文 (Ye Shi-wen) is a proper noun, and it is likely to get an incorrect segmentation. If we use the wrong result of segmentation to build the syntax tree, the tree is also incorrect, as illustrated in Figure 5.1b. Comparing to the correct syntax tree in Figure 5.1a, the head words and the goal/theme NP subtrees are different in both trees. It shows that a minor error in segmentation can have a larger impact while building syntax tree. The input of the CKIP Parser accepts both unsegmented text and segmented text. If we want to reduce the error of segmentation, it is also possible to use other application for segmentation or manually do the segmentation.

Error in Syntax Tree Construction

In some cases, although the result of segmentation is correct, however, the syntax tree is wrong. There is lots of ambiguities in the grammar of natural language; hence, it



(a) Syntax Tree after the Correct Segmentation



(b) Syntax Tree after the Incorrect Segmentation

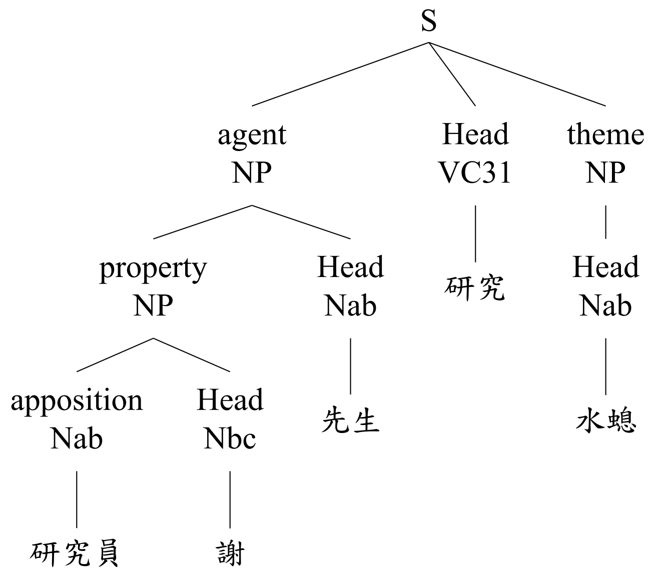
Figure 5.1: Syntax Tree after Chinese Word Segmentation

is possible to generate multiple grammatical-correct syntax trees from a single sentence. Furthermore, in Chinese, the morphology of verb, noun and adjective are all the same. As a result, it is likely to mistake a noun for a verb. For example, the sentence shown in (5.4) can be correctly segmented, and the result of segmentation is shown in (5.5).

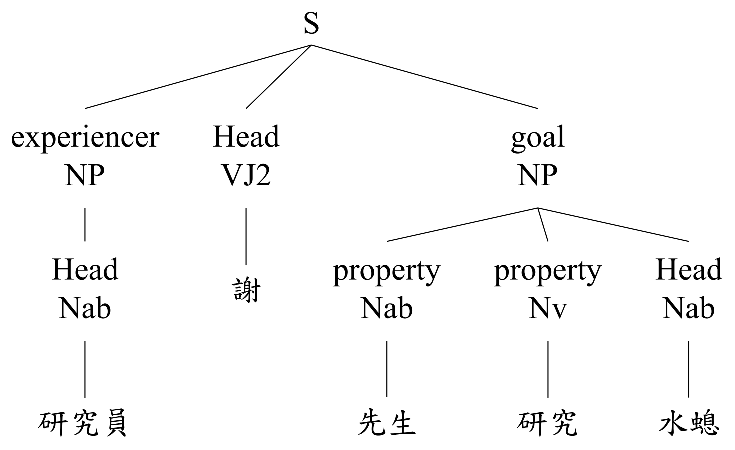
研究員謝先生研究水螅 (5.4)

研究員｜謝｜先生｜研究｜水螅 (5.5)

However, the syntax tree constructed by the CKIP Parser is wrong, as illustrated in Figure 5.2b. Comparing to the correct syntax tree shown in Figure 5.2a, the CKIP Parser mistakes the word 謝 (Hsieh) for a verb, and regards it as the head word of the sentence. Actually, this word is a noun rather than a verb. Furthermore, in the syntax tree, it is the sibling of 研究員 (researcher). In this case, the CKIP Parser constructs the wrong syntax tree in Figure 5.2b, and its structure is different from the correct result in Figure 5.2a. In



(a) Correct Syntax Tree



(b) Incorrect Syntax Tree

Figure 5.2: Example of Correct/Incorrect Syntax Trees

order to reduce the parsing error, it is possible to use other Chinese parser to validate the result of the CKIP Parser. However, in this work, we do not intend to do it.

5.1.2 Restriction on Data Size

The CKIP Parser also has some restrictions on the size of input data. CKIP Parser can't process the sentence with more than 30 Chinese words. Also, the online version of CKIP Parser cannot process the document with more than 20 sentences. However, in real-world application, it is common to have a sentence with more than 30 words, or a document with more than 20 sentences. These restrictions restrict the possibility of using the CKIP Parser in application of Big Data with the scale of megabytes or gigabytes.

5.2 Semantic Constructor

In this work, we present an algorithm to build semantic representation from syntax tree. However, some linguistic phenomena are outside the capability of our Semantic Constructor, including quantifier, anaphora, etc. We need more advance theory of formal semantics and more complex algorithm to construct the semantic representations of these phenomena. In this work, we only focus on simple RTE implementation, and these phenomena are outside the scope of this thesis.

5.2.1 Quantifier

Quantifiers are the operators used to bind the variable's reference to some set of individuals. Two basic operators in first-order logic are the existential quantifier and the universal quantifier. In natural language, we can use quantifiers to express the semantics of the words that indicate quantity. For example, (5.6) contains the words 所有 (all) and 有些 (some).

所有 (all) 學生 (student) 通過 (pass) 考試 (exam) (5.6a)

有些 (some) 學生 (student) 通過 (pass) 考試 (exam) (5.6b)

It is possible to use quantifiers to express the semantics of (5.6a) and (5.6b), shown in (5.7a) and (5.7b) respectively.

$$\forall x.(\text{學生}(x) \wedge agent(x,e)) \rightarrow (\text{通過}(e) \wedge goal(e,y) \wedge \text{考試}(y)) \tag{5.7a}$$

$$\exists x.(\text{學生}(x) \wedge agent(x,e)) \wedge (\text{通過}(e) \wedge goal(e,y) \wedge \text{考試}(y)) \tag{5.7b}$$

However, when there are multiple quantifiers in a sentence, the scopes of these quantifiers become ambiguous. For example, in (5.8), there are two quantifiers: 每個 (every) and 一些 (some).

$$\text{每個 (every) 學生 (student) 都有 (have) 一些 (some) 作業 (homework)} \tag{5.8}$$

There are two possible semantic representations for this sentence, as shown in (5.9).

$$\forall x \exists y.(\text{學生}(x) \land agent(x,e)) \rightarrow (\text{有}(e) \land goal(e,y) \land \text{作業}(y)) \tag{5.9a}$$

$$\exists y \forall x.(\text{學生}(x) \land agent(x,e)) \rightarrow (\text{有}(e) \land goal(e,y) \land \text{作業}(y)) \tag{5.9b}$$

In (5.9a), all the students can have different homework. On the other hand, in (5.9b), all the students have the same homework. Although it is possible to solve the quantifier ambiguity by other advanced techniques, however, it is outside the scope of this thesis.

5.2.2 Anaphora

Anaphora is an expression that depends upon another expression in context. For example, in (5.10), the pronoun 他 (he) is an anaphoric expression.

$$\text{小明 (XiaoMing) 說 (say) 他 (he) 通過 (pass) 考試 (exam)} \tag{5.10}$$

According to this sentence, we can infer that the word 他 (he) refers to the person 小明 (XiaoMing). However, it is more difficult to resolve the anaphora if there are several possible candidates in the context. For example, in (5.11), the pronoun 他 (he) may be referred to the person 小明 (XiaoMing) or 小華 (XiaoHwa).

$$\begin{array}{l}\text{小華 (XiaoHwa) 看起來 (look) 很 (very) 高興 (happy)} \\ \text{小明 (XiaoMing) 說 (say) 他 (he) 通過 (pass) 考試 (exam)}\end{array} \tag{5.11}$$

It requires additional information to solve the anaphoric phrase, such as syntactic information and common-sense knowledge. Also, it is outside the scope of this thesis.

5.3 RTE Engine

In this work, we use the Knowledge Builder to build the logical forms of required knowledge from input logical forms, and we use the Knowledge Validator to validate the correctness of the required knowledge. However, there are many cases whose logical forms of required knowledge can't be built, or their required knowledge can't be correctly validated. We will elaborate on these situations in the following paragraphs.

5.3.1 Knowledge Builder

Since we use an extremely simple algorithm to build the logical forms of required knowledge from input logical forms, however, in several cases, the logical forms of required knowledge can't be built by our algorithm. We give some examples of these situations and explain why our algorithm is unable to build them.

Building Knowledge from Different Semantic Roles

Although the thematic roles from the CKIP Parser enable the construction of Neo-Davidsonian semantic representation, in some cases, the mismatch of thematic roles between T and H makes the Knowledge Builder unable to build the logical form of required knowledge. For example, in (5.12), the syntax trees of this example are shown in Figure 5.3.

$$\begin{aligned} T&: \text{小明玩電腦遊戲} \\ H&: \text{小明玩電腦} \end{aligned} \tag{5.12}$$

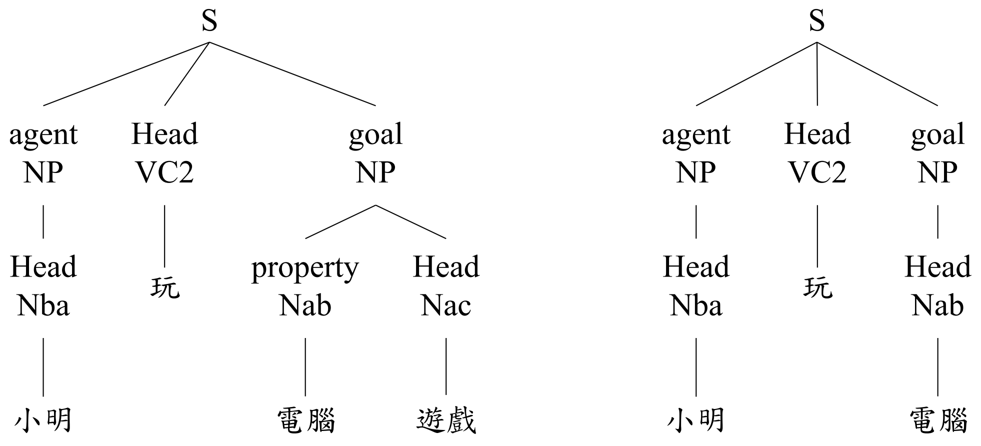
Then, we use these syntax trees to build logical forms in (5.13).

$$\begin{aligned} T:\ & \text{小明}(n_0) \wedge agent(n_0, e) \wedge \text{玩}(e) \wedge \text{電腦}(n_1) \wedge property(n_1, n_2) \\ & \wedge\text{遊戲}(n_2) \wedge goal(n_2, e) \\ H:\ & \text{小明}(n_0) \wedge agent(n_0, e) \wedge \text{玩}(e) \wedge \text{電腦}(n_1) \wedge goal(n_1, e) \end{aligned} \tag{5.13}$$

If we use the algorithm in the Knowledge Builder to construct the logical forms, we will get the logical form K in (5.14).

$$K : \text{遊戲}(n_2) \wedge goal(n_2, e) \rightarrow \text{電腦}(n_1) \wedge goal(n_1, e) \tag{5.14}$$

In fact, the word 電腦 (computer) in H should be aligned to the word 電腦 in T, and should not be aligned to the word 遊戲 (game) in T. However, the CKIP Parser will assign different roles to same word 電腦.



(a) Syntax Tree of T　　(b) Syntax Tree of H

Figure 5.3: Syntax Trees of Example (5.12)

Building Knowledge from Multiple Words

In some cases, the synonym pair between T and H may span to multiple words. For example, in (5.15), the synonyms between T and H are 電腦遊戲 (computer game) and 電玩 (computer game).

$$\begin{aligned} T &: \text{小明玩電腦遊戲} \\ H &: \text{小明玩電玩} \end{aligned} \tag{5.15}$$

However, if we use the CKIP Parser to build the syntax tree, and convert it into logical form (5.16), the word 電腦遊戲 (computer game) is split into two words: 電腦 (computer) and 遊戲 (game).

$$\begin{aligned} T:\ & 小明(n_0) \wedge agent(n_0, e) \wedge 玩(e) \wedge 電腦(n_1) \wedge property(n_1, n_2) \\ & \wedge 遊戲(n_2) \wedge goal(n_2, e) \\ H:\ & 小明(n_0) \wedge agent(n_0, e) \wedge 玩(e) \wedge 電玩(n_3) \wedge goal(n_3, e) \end{aligned} \quad (5.16)$$

If we use the algorithm in the Knowledge Builder to construct the logical form of required knowledge, we will get the logical form K in (5.17). It only contains 遊戲 in T.

$$K:\ 遊戲(n_2) \wedge goal(n_2, e) \rightarrow 電玩(n_3) \wedge goal(n_3, e) \quad (5.17)$$

However, the logical form of required knowledge K' should contain both words 電腦 and 遊戲 in T, as shown in (5.18).

$$K':\ 電腦(n_1) \wedge property(n_1, n_2) \wedge 遊戲(n_2) \wedge goal(n_2, e) \rightarrow 電玩(n_3) \wedge goal(n_3, e) \quad (5.18)$$

Building Knowledge of Non-Action Verbs

For some non-action verbs, they have special meaning or function in the sentence. For example, in (5.19), this example contains the word 是 (is), and it represents the state of being.

T: 學生王小明在玩電腦

H: 王小明是學生 (5.19)

This example can be converted into logical form in (5.20).

$$\begin{aligned} T:\ & 學生(n_0) \wedge apposition(n_0, n_1) \wedge 王小明(n_1) \wedge agent(n_1, e) \wedge 玩(e) \\ & \wedge 電腦(n_2) \wedge goal(n_2, e) \\ H:\ & 王小明(n_1) \wedge theme(n_1, e) \wedge 是(e) \wedge 學生(n_0) \wedge range(n_0, e) \end{aligned} \quad (5.20)$$

In this case, we require the logical form of knowledge K in (5.21) to solve the entailment problem.

$$\begin{aligned} K: \ & \text{學生}(n_0) \wedge apposition(n_0, n_1) \wedge \text{王小明}(n_1) \rightarrow \\ & \text{王小明}(n_1) \wedge theme(n_1, e) \wedge \text{是}(e) \wedge \text{學生}(n_0) \wedge range(n_0, e) \end{aligned} \tag{5.21}$$

However, this logical form can't be easily built, we do not intend to implement the algorithm to build these kinds of logical form in our system.

Building Knowledge of Complex Concept

In some real-world cases, the structure of syntax trees can be very complex. To build the logical form of required knowledge, we need to consider the relationship spreading between several nodes in the syntax trees. Since the logical form is constructed from syntax trees, the logical form can be very complex. As example in (5.22), the syntax trees of this example are shown in Figure 5.4.

$$\begin{aligned} T: \ & \text{今年世足賽在聖保羅競技場開響} \\ H: \ & \text{今年世足賽第一場比賽在聖保羅競技場舉行} \end{aligned} \tag{5.22}$$

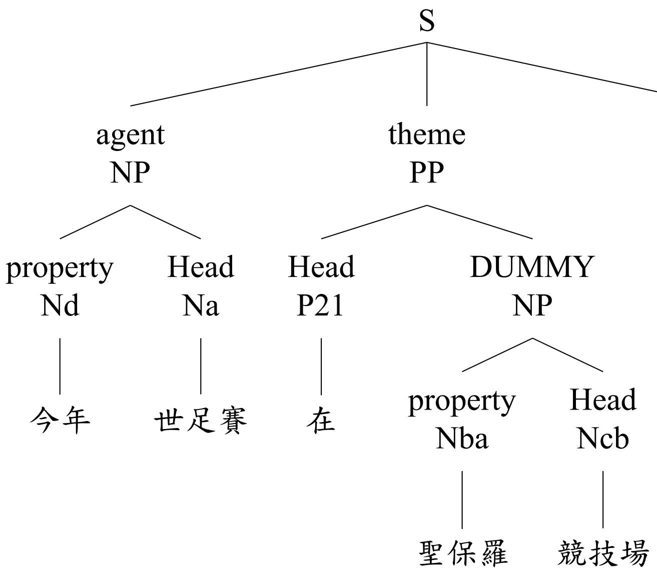
We build the logical forms of this example, as shown in (5.23).

$$\begin{aligned} T: \ & \text{今年}(n_0) \wedge property(n_0, n_2) \wedge \text{世足賽}(n_2) \wedge agent(n_2, e) \wedge \text{聖保羅}(n_1) \\ & \wedge property(n_1, n_3) \wedge \text{競技場}(n_3) \wedge theme(n_3, e) \wedge \text{開響}(e) \\ H: \ & \text{今年}(n_0) \wedge property(n_0, n_4) \wedge \text{世足賽}(n_2) \wedge property(n_2, n_4) \\ & \wedge \text{第一場}(n_5) \wedge property(n_5, n_4) \wedge \text{比賽}(n_4) \wedge agent(n_4, e) \wedge \text{聖保羅}(n_1) \\ & \wedge property(n_1, n_3) \wedge \text{競技場}(n_3) \wedge location(n_3, e) \wedge \text{舉行}(e) \end{aligned} \tag{5.23}$$

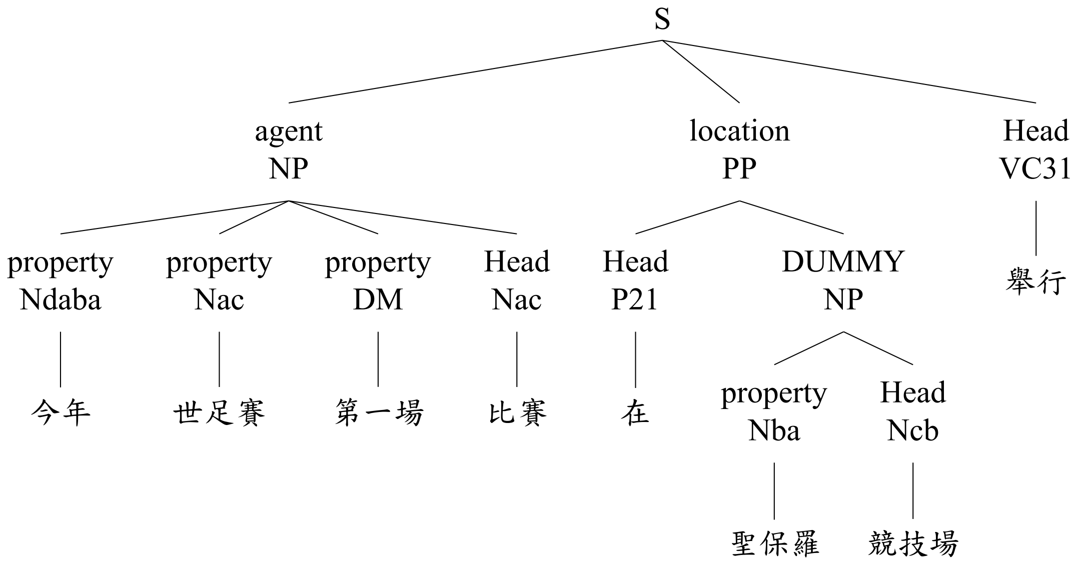
In this case, we need to add the logical form of external knowledge K (5.24), about the concept between the words 開響, 舉行 (hold), 第一場 (first) and 比賽 (game).

$$K: \text{開響}(e) \rightarrow \text{舉行}(e) \wedge \text{第一場}(n_5) \wedge property(n_5, n_4) \wedge \text{比賽}(n_4) \wedge agent(n_4, e). \tag{5.24}$$

However, the algorithm in the Knowledge Builder cannot build such complex knowledge, we need to manually build it. Moreover, in this case, the thematic roles of the word 世足賽 are different in H and in T, so we have the problem of *building knowledge from different semantic roles*, discussed in the previous section.



(a) Syntax Tree of T



(b) Syntax Tree of H

Figure 5.4: Syntax Trees of Example (5.22)

5.3.2 Knowledge Validator

It is possible that the Knowledge Validator will validate the incorrect knowledge in the logical forms of required knowledge. Also, the Knowledge Validator may drop the correct one. In the following sections, we discuss the situations when the Knowledge Validator makes mistakes. Finally, we discuss the reason why ontology is not adopted as the knowledge resource in this work.

Chinese WordNet

There are some situations that our Knowledge Validator will make mistakes when using the Chinese WordNet[57]. Also, the current version of Chinese WordNet has its limitation while using it.

Multiple Word Senses In this work, the function in the Knowledge Validator, $checkCwn$, checks the relations of synonym and hypernym from all senses of $w1$. Since a word can have multiple senses, it is possible to choose the wrong sense. For example, to solve the problem in (5.25), we need to check whether 打 (play) and 揍 (beat) is synonym or not.

$$\begin{aligned} &T: \text{小明 (XiaoMing) 打 (play) 球 (ball)} \\ &H: \text{小明 (XiaoMing) 揍 (beat) 球 (ball)} \\ &T \rightarrow H = FALSE \end{aligned} \tag{5.25}$$

The word 打 has multiple senses. In this example, the sense of 打 is *play*, and 揍 is not the synonym or hypernym of this sense. However, the word 打 has another sense *beat*, and 揍 (beat) is the synonym of this sense. In this case, the function $checkCwn$ will return true, making the knowledge of this logical form valid. However, the actual validity of this knowledge is false. Although it is possible to choose the correct sense by *word-sense disambiguation* [61], however, in the research of natural language processing, it is a difficult task. In this work, we did not implement the process of word-sense disambiguation.

Coverage of Chinese WordNet It is possible to find a word that is not in the Chinese WordNet. For example, the word 電玩 (computer game) is not in the Chinese WordNet. In this situation, it does not mean that the knowledge is invalid and we should not drop out the logical form of this knowledge, since the coverage of the Chinese WordNet is very low. In this work, we use the Chinese LSA Website to handle this situation. It is also possible to use other resources of lexical semantics with wider coverage.

Chinese LSA Website

Since the semantic similarity from the Chinese LSA Website[58] is calculated by cosine similarity. It is not robust enough to decide whether the semantic relation between a pair of words is synonym, hypernym or not. It is possible that a pair of words with similar context but different meaning will have a higher value of cosine similarity. For example, the cosine similarity between the word 老師 (teacher) and the word 學生 (student) is 0.301413361635, higher than the cosine similarity between the word 老師 (teacher) and the word 教師 (instructor), whose value is 0.291650605182. However, the semantic relation between 老師 and 教師 is synonym, but the meaning of 老師 and 學生 is different. Hence, it is possible to make a mistake when taking 老師 and 學生 as synonym by cosine similarity. In this work, we only use the Chinese LSA Website. It is possible to use larger corpus to get more robust result, but it takes more computational resources. Also, it is possible to choose different measurement of semantic similarity other than cosine similarity.

Ontology

There are some reasons why we do not adopt ontology as the knowledge resource. First, although ontology can be converted into description logic, however, in order to integrate these logical forms into the knowledge built by the Knowledge Builder, we need to convert description logic into the semantic representation of Neo-Davidsonian semantics. Second, since ontology is a knowledge resource of world knowledge, not lexical knowledge, hence, we need a mapping to map every lexicon to ontology. As a result, we do not use ontology in this work. However, in the future work, it is possible to integrate ontology into our framework.

Chapter 6

Experiment, Result and Discussion

The purpose of this experiment is not going to show that our system is able to compete with other existing methods, such as machine learning, but to demonstrate the capabilities and limitations of our system by giving some examples of how our system can (or cannot) solve the RTE problems. This experiment can be categorized into two types: experiment on simple test cases and the RITE test cases. The simple test cases are the test cases manually created, and the RITE test cases are the test cases from the NTCIR RITE Competition [50]. These test cases are created from real-world text, such as digital newswire corpus. After running these test cases, we discuss the reason why our system is able to (or fail to) solve these problems.

6.1 Experiment on Simple Test Cases

The evaluation of computational semantics in English can be done by the FRACAS test suite. The data in the FRACAS test suite is designed for evaluating semantic representations, and can minimize the errors in the stages before semantic construction, such as syntactic parsing. However, there is no such test case for computational semantics in Chinese. Hence, we manually create the test cases for this purpose. The test cases are created to test the different situations that our semantic constructor and RTE Engine can (or can't) handle.

6.1.1 Simple Test Cases: Success

We manually create the test cases to demonstrate that these test cases can be solved by our system successfully. The test cases are shown in Table 6.1. In this table, each

ID	Text		T1 entails T2	T2 entails T1
Basic Case				
1	T1	柯文哲今天發表演講	True	False
	T2	柯文哲發表演講		
Prepositional Phrase, Noun Phrase and DE Phrase				
2a	T1	柯文哲今天發表演講	True	True
	T2	柯文哲在今天發表演講		
2b	T1	柯文哲發表精彩演講	True	True
	T2	柯文哲發表精彩的演講		
Embedded Sentence				
3	T1	柯文哲希望同學建立自己的人生哲學	True	False
	T2	柯文哲希望同學建立人生哲學		
Negation				
4a	T1	柯文哲沒有發表演講	True	False
	T2	柯文哲今天沒有發表演講		
4b	T1	柯文哲今天沒有不發表演講	True	True
	T2	柯文哲今天發表演講		
Coordination				
5a	T1	柯文哲在昨天和今天發表演講	True	False
	T2	柯文哲在今天發表演講		
5b	T1	柯文哲在今天發表演講	True	False
	T2	柯文哲在昨天或今天發表演講		
External Knowledge				
6a	T1	柯文哲發表演講	True	True
	T2	柯文哲發表演說		
6b	T1	柯文哲醫生發表演講	True	True
	T2	柯文哲醫師發表演講		

Table 6.1: Simple Test Cases: Success

test case contains two text fragments: T1 and T2. We want to use our system to check whether it can prove the entailment relationship between T1 and T2 or not. The golden results of whether T1 entails T2 and T2 entails T2 are listed in the last two columns. Case 1 is the Basic Case. The logical form of this case can be created by the basic semantic constructing algorithm in Section 4.3.1, without any special treatments of Prepositional Phrase, Negation or Coordination. The other cases contain some phenomena for special treatments to constructing the semantics, including prepositional phrase, noun phrase, DE phrase, embedded sentence, negation and coordination. Also, some of these cases require external knowledge to be solved. We will elaborate on how our system can solve these test cases in the following paragraphs.

Basic Case

Case 1 is the Basic Case. This case can be solved by converting it into the logical forms in (6.1).

$$\begin{aligned} T1:\ & \text{柯文哲}(n_0) \wedge agent(n_0, e) \wedge \text{今天}(n_1) \wedge time(n_1, e) \wedge \text{發表}(e) \\ & \wedge \text{演講}(n_2) \wedge theme(n_2, e) \end{aligned} \tag{6.1a}$$

$$T2:\ \text{柯文哲}(n_0) \wedge agent(n_0, e) \wedge \text{發表}(e) \wedge \text{演講}(n_2) \wedge theme(n_2, e) \tag{6.1b}$$

Then, we can use the theorem prover to prove that $T1 \rightarrow T2$ is true and $T2 \rightarrow T1$ is false.

Prepositional Phrase, Noun Phrase and DE Phrase

Case 2a contains the prepositional word 在 (in) in $T2$. In Chinese, the word 在 (in) can be ignored in this case. We use the method in Section 4.3.2 to convert both $T1$ and $T2$ into logical forms, which are the same as $T1$ in (6.1a). Then, the theorem prover solves that $T1 \rightarrow T2$ and $T2 \rightarrow T1$ is true.

Case 2b contains the word 的 (DE) in $T2$. In Chinese, the word 的 (DE) can also be ignored in this case. Hence, the semantic representations of $T1$ and $T2$ are the same.

Embedded Sentence

Case 3 contains embedded sentences in both $T1$ and $T2$. We use the method in Section 4.3.5 to construct the logical forms of sentences with embedded sentence. In this case, we have two different verbs: 希望 (hope) and 建立 (establish), and the variables for their events are e_1 and e_2 respectively. The logical forms of this case are shown in (6.2). After constructing the logical forms, we can use the theorem prover to solve this case.

$$T1: \text{柯文哲}(n_0) \wedge experiencer(n_0, e) \wedge \text{希望}(e) \wedge \text{同學}(n_3) \wedge agent(n_3, e_2) \wedge \text{建立}(e_2) \wedge \text{自己}(n_1) \wedge possessor(n_1, n_4) \wedge \text{人生}(n_2) \wedge property(n_2, n_4) \wedge \text{哲學}(n_4) \wedge theme(n_4, e_2) \wedge goal(e_2, e) \quad (6.2a)$$

$$T2: \text{柯文哲}(n_0) \wedge experiencer(n_0, e) \wedge \text{希望}(e) \wedge \text{同學}(n_3) \wedge agent(n_3, e_2) \wedge \text{建立}(e_2) \wedge \text{人生}(n_2) \wedge property(n_2, n_4) \wedge \text{哲學}(n_4) \wedge theme(n_4, e_2) \wedge goal(e_2, e) \quad (6.2b)$$

Negation

Case 4a contains the negation word 沒有 (not) in both $T1$ and $T2$. We can use the method introduced in Section 4.3.6 to convert the sentences with negation words into logical forms, as shown in (6.3). Since (6.3a) is the negation of (6.1b), and (6.3b) is the negation of (6.1a). We also know that, if $A \rightarrow B$ is true, $\neg B \rightarrow \neg A$ is also true. So, in this case, $T1 \rightarrow T2$ is true, and we can use the theorem prover to prove that.

$$T1: \neg(\text{柯文哲}(n_0) \wedge agent(n_0, e) \wedge \text{發表}(e) \wedge \text{演講}(n_2) \wedge theme(n_2, e)) \quad (6.3a)$$

$$T2: \neg(\text{柯文哲}(n_0) \wedge agent(n_0, e) \wedge \text{今天}(n_1) \wedge time(n_1, e) \wedge \text{發表}(e) \wedge \text{演講}(n_2) \wedge theme(n_2, e)) \quad (6.3b)$$

Case 4b contains double negation words 沒有 (not) and 不 (not) in $T1$, so they cancel each other. The resulting logical form of $T1$ is the same as $T2$.

Coordination

Case 5a contains the coordinating word 和 (and) in $T1$. The meaning of 和 (and) is conjunction, so the semantic representation of the word 和 (and) is the logical operator $\wedge$, and the logical form of $T1$ is in (6.4). The logical form of $T2$ is the same as (6.1a), and we can use the theorem prover to solve that $T1 \rightarrow T2$ is true, and $T2 \rightarrow T1$ is false.

$$\begin{aligned} T1:\ & \text{柯文哲}(n_0) \wedge agent(n_0, e) \wedge \text{昨天}(n_3) \wedge time(n_3, e) \wedge \text{今天}(n_1) \\ & \wedge time(n_1, e) \wedge \text{發表}(e) \wedge \text{演講}(n_2) \wedge theme(n_2, e) \end{aligned} \tag{6.4}$$

Case 5b contains the coordinating word 或 (or) in $T2$. The meaning of 或 (or) is disjunction, we can use the same method in **Case 5a** to solve this problem, and the semantic representation of 或 (or) is the logical operator $\vee$. The resulting logical form is in (6.5).

$$\begin{aligned} T2:\ & \text{柯文哲}(n_0) \wedge agent(n_0, e) \wedge ((\text{昨天}(n_3) \wedge time(n_3, e)) \vee (\text{今天}(n_1) \\ & \wedge time(n_1, e))) \wedge \text{發表}(e) \wedge \text{演講}(n_2) \wedge theme(n_2, e) \end{aligned} \tag{6.5}$$

External Knowledge

Case 6a requires external knowledge. We construct the logical forms of $T1$ and $T2$. $T1$ is the same as (6.1b), and $T2$ is in (6.6), respectively.

$$T2:\ \text{柯文哲}(n_0) \wedge agent(n_0, e) \wedge \text{發表}(e) \wedge \text{演說}(n_3) \wedge theme(n_3, e) \tag{6.6}$$

The Knowledge Builder builds the logical forms of required knowledge of this case. If we want to prove that $T1$ entails $T2$, we need (6.7a). If we want to prove that $T2$ entails $T1$, we need (6.7b).

$$K1:\ \text{演講}(n_2) \wedge theme(n_2, e) \rightarrow \text{演說}(n_3) \wedge theme(n_3, e) \tag{6.7a}$$

$$K2:\ \text{演說}(n_3) \wedge theme(n_3, e) \rightarrow \text{演講}(n_2) \wedge theme(n_2, e) \tag{6.7b}$$

Both of them can be validated by the Knowledge Validator. The Knowledge Validator can find that the word 演講 (speech) is a synonym of the word 演說 (speech) in the Chinese WordNet. Hence, both $K1$ and $K2$ are valid.

Case 6b is similar to the **Case 6a**. The Knowledge Validator needs to validate the required knowledge by searching whether the words 醫生 (doctor) and 醫師 (doctor) are synonym or not. However, the semantic relation between these two words are not in the Chinese WordNet, so we need to calculate their semantic similarity from the Chinese LSA Website. The result is 0.619 and that is higher than the threshold 0.1. Therefore, the required knowledge is valid.

6.1.2 Simple Test Cases: Failure

We manually create the test cases to show that our system can't solve these test cases successfully. These test cases are shown in Table 6.2. Due to the reasons discussed in Chapter 5. Our system may fail to solve the problems due to several reasons, including errors or exceptions in the CKIP Parser, Semantic Constructor, Knowledge Builder and Knowledge Validator. We will elaborate on how our system failed when solving these test cases.

ID	Text		T1 entails T2	T2 entails T1
Error in the CKIP Parser				
1	T1	同學有聽醫師柯文哲發表演講	True	False
	T2	同學有聽柯文哲發表演講		
Exception in Semantic Constructor				
2a	T1	所有同學都來聽演講	True	False
	T2	所有大一同學都來聽演講		
2b	T1	柯文哲說他要參選	True	True
	T2	柯文哲說自己要參選		
Exception in Knowledge builder				
3	T1	柯文哲是台大附設醫院的醫師	True	False
	T2	柯文哲是教學醫院的醫師		
Error in Knowledge Validator				
4a	T1	柯文哲打這場選戰	False	False
	T2	柯文哲評這場選戰		
4b	T1	柯文哲是醫師	False	False
	T2	柯文哲是護士		

Table 6.2: Simple Test Cases: Failure

Error in the CKIP Parser

Case 1 cannot be parsed correctly by the CKIP Parser. The resulting syntax tree of $T1$ in this case is in Figure 6.1a. This syntax tree is wrong due to some reasons. First, the segmentation is wrong because the proper name 柯文哲 (Ko Wen-Je) is split into two tokens: 柯 (Ko) and 文哲 (Wen-Je). Second, the third child and fourth child of node S should be merged into a child. The correct syntax tree is illustrated in Figure 6.1b.

Exception in Semantic Constructor

Case 2a contains the word 所有 (all), the semantic representation of this word is the quantifier $\forall$. However, we have no algorithm to construct quantifier (see Section 5.2.1) in logical form. The theorem prover cannot prove that $T1$ entails $T2$.

Case 2b contains the pronouns 他 (he) and 自己 (-self), and these pronouns will result in anaphora. The logical forms constructed by Semantic Constructor did not express who refer to the pronouns 他 (he) and 自己 (-self). Hence, the theorem prover can't prove the entailment in this case.

Exception in Knowledge Builder

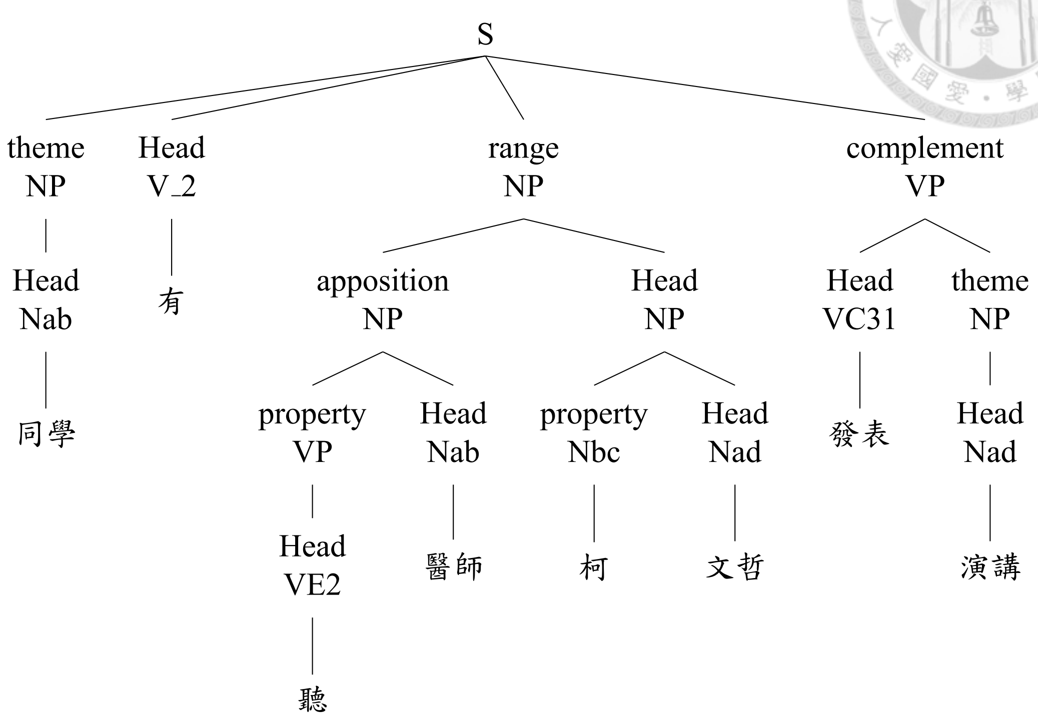
Case 3 contains two noun phrases: 台大 (NTU) 附設 (affiliated) 醫院 (hospital) and 教學 (Teaching) 醫院 (hospital). The logical forms of this case are in (6.8).

$$\begin{aligned} T1:\ & \text{柯文哲}(n_0) \wedge time(n_0, e) \wedge \text{是}(e) \wedge \text{台大}(n_2) \wedge property(n_2, n_3) \wedge \text{附設}(n_3) \\ & \wedge \text{醫院}(n_1) \wedge theme(n_1, n_3) \wedge predication(n_3, n_5) \wedge \text{醫師}(n_5) \wedge range(n_5, e) \end{aligned} \tag{6.8a}$$

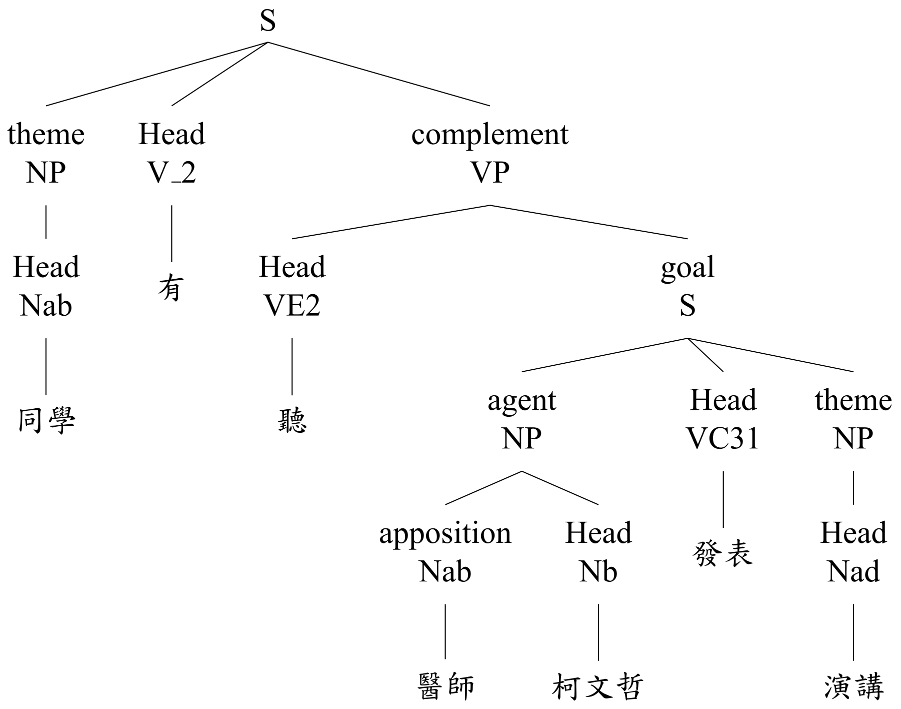
$$\begin{aligned} T2:\ & \text{柯文哲}(n_0) \wedge time(n_0, e) \wedge \text{是}(e) \wedge \text{教學}(n_6) \wedge property(n_6, n_1) \wedge \text{醫院}(n_1) \\ & \wedge property(n_1, n_5) \wedge \text{醫師}(n_5) \wedge range(n_5, e) \end{aligned} \tag{6.8b}$$

To solve this case, we require the logical form of knowledge in (6.9) to establish the entailment between the two noun phrases. However, as discussed in Section 5.3.1, our Knowledge Builder is unable to build the logical form of knowledge of noun phrases containing multiple words.

$$\begin{aligned} & \text{台大}(n_2) \wedge property(n_2, n_3) \wedge \text{附設}(n_3) \wedge \text{醫院}(n_1) \wedge theme(n_1, n_3) \\ & \rightarrow \text{教學}(n_6) \wedge property(n_6, n_1) \wedge \text{醫院}(n_1) \end{aligned} \tag{6.9}$$



(a) Wrong Syntax Tree



(b) Correct Syntax Tree

Figure 6.1: Syntax Tree of T1 in Case 1

Error in Knowledge Validator

Case 4a needs extra knowledge to prove the truth of entailment. The Knowledge Builder builds the logical form of required knowledge of this case, as shown in (6.10).

$$K: \ \text{打}(e) \rightarrow \text{評}(e) \tag{6.10}$$

The Knowledge Validator can validate this entailment because one of the senses in lemma 打 is synonym of the word 評 (rate). Thus, the theorem prover will prove the entailment between $T1$ and $T2$ is true. However, the sense of 打 in this sentence is *fight*, which is different from the sense of 評 (rate). This knowledge should be invalidated, and the entailment between $T1$ and $T2$ should be false.

Case 4b is in the similar situation as in **Case 4a**. The required knowledge that the word 醫生 (doctor) entails 護士 (nurse) should be invalidated. However, the Knowledge Validator will validate this knowledge by the Chinese LSA Website. Since the semantic similarity between 醫生 (doctor) and 護士 (nurse) is 0.67, which is higher than the threshold 0.1.

6.2 Experiment on RITE Competition Test Cases

We use the test cases of Traditional Chinese Multi-class subtask in the NTCIR RITE Competition [50]. The test cases of RITE are obtained from real-world text. The multi-class classification is to classify a pair of text $T1$ or $T2$ into one of the five categories considering entailment direction, paraphrase and contradiction. The labels in the MC test cases are the following:

F: forward entailment ($T1$ entails $T2$ and $T2$ does not entail $T1$).

R: reverse entailment ($T2$ entails $T1$ and $T1$ does not entail $T2$).

B: bidirectional entailment ($T1$ entails $T2$ and $T2$ entails $T1$).

C: contradiction ($T1$ and $T2$ contradict, or cannot be true at the same time).

I: independence (otherwise)

To classify whether the label of a test case is **F**, **R** and **B**, we use the theorem prover to prove whether $T1 \rightarrow T2$ and $T2 \rightarrow T1$. To classify whether the label is **C** or **I**, we prove whether $T1 \rightarrow \neg T2$. If the result is true, the label is **C**; otherwise, the label is **I**.

6.2.1 RITE Test Cases: Success

This experiment is not designed to show that what percentage of test cases can be solved correctly, but to demonstrate the procedures of how some test cases can be successfully solved by our system. These cases are listed in Table 6.3. The column ID is the class id in NCTIR RITE dataset. We categorize them into four classes: Without External Knowledge, With External Knowledge, Negation and Independence.

ID	Text		Label
Without External Knowledge			
130	T1	卡塔米主張開放人權和民主改革	F
	T2	卡塔米主張民主改革	
With External Knowledge			
66	T1	安南出身於非洲迦納	B
	T2	安南來自非洲迦納	
Negation			
45	T1	小泉純一郎 2001 年贏得自民黨總裁選戰	C
	T2	小泉純一郎 2001 年未贏得自民黨總裁選戰	
Independence			
4	T1	思科公司是全球最大的網路供應公司	I
	T2	微軟是全球最大軟體公司	

Table 6.3: RITE Test Cases: Success

Without External Knowledge

Case 130 contains the coordinating word 和 (and) in $T1$, and we can build the semantics of 和 (and) as conjunctive operator $\wedge$. The resulting logical form is in (6.11). Then, the theorem prover can prove that $T1$ entails $T2$ and $T2$ does not entails $T1$; hence, this case can be classified as **F**.

$$\begin{aligned} T1:\ & \text{卡塔米}(n_2) \wedge agent(n_2, e) \wedge \text{主張}(e) \wedge \text{開放}(n_3) \wedge \text{人權}(n_0) \wedge goal(n_0, n_3) \\ & \wedge goal(n_3, e) \wedge \text{民主}(n_1) \wedge property(n_1, n_4) \wedge \text{改革}(n_4) \wedge goal(n_4, e) \end{aligned} \tag{6.11a}$$

$$T2: \text{卡塔米}(n_2) \wedge agent(n_2, e) \wedge \text{主張}(e) \wedge \text{民主}(n_1) \wedge property(n_1, n_4) \\ \wedge \text{改革}(n_4) \wedge goal(n_4, e) \tag{6.11b}$$

With External Knowledge

Case 66 requires external knowledge. First, we build the logical forms of both $T1$ and $T2$, as shown in (6.12).

$$T1: \text{安南}(n_1) \wedge theme(n_1, e) \wedge \text{出身}(e) \wedge \text{非洲}(n_0) \wedge property(n_0, n_2) \\ \wedge \text{迦納}(n_2) \wedge range(n_2, e) \tag{6.12a}$$

$$T2: \text{安南}(n_1) \wedge theme(n_1, e) \wedge \text{來自}(e) \wedge \text{非洲}(n_0) \wedge property(n_0, n_2) \\ \wedge \text{迦納}(n_2) \wedge range(n_2, e) \tag{6.12b}$$

Then, the Knowledge Builder creates the logical forms of required knowledge $K1$ and $K2$ in (6.13).

$$K1: \text{出身}(e) \rightarrow \text{來自}(e) \tag{6.13a}$$

$$K2: \text{來自}(e) \rightarrow \text{出身}(e) \tag{6.13b}$$

The Knowledge Validator can validate both (6.13a) and (6.13b) from the Chinese LSA Website, and the semantic similarity between 出身 (originated from) and 來自 (come from) is 0.184, which is higher than the threshold 0.1. The theorem prover can prove $T1 \wedge K1$ entails $T2$, and $T2 \wedge K2$ entails $T1$, and this case can be classified as **B**.

Negation

Case 45 contains a negation word in $T2$. When we convert it into logical form, the semantics should be negated. The logical forms of this case are in (6.14).

$$T1: \text{小泉}(n_0) \wedge property(n_0, n_3) \wedge \text{純一郎}(n_3) \wedge theme(n_3, e) \wedge 2001\text{ 年}(n_4) \\ \wedge time(n_4, e) \wedge \text{贏得}(e) \wedge \text{自民黨}(n_1) \wedge property(n_1, n_5) \wedge \text{總裁}(n_2) \\ \wedge property(n_2, n_5) \wedge \text{選戰}(n_5) \wedge range(n_5, e) \tag{6.14a}$$

$$\begin{aligned} T2:\ & \neg(\text{小泉}(n_0) \wedge property(n_0, n_3) \wedge \text{純一郎}(n_3) \wedge theme(n_3, e) \wedge 2001\ \text{年}(n_4) \\ & \wedge time(n_4, e) \wedge \text{贏得}(e) \wedge \text{自民黨}(n_1) \wedge property(n_1, n_5) \wedge \text{總裁}(n_2) \\ & \wedge property(n_2, n_5) \wedge \text{選戰}(n_5) \wedge range(n_5, e)) \end{aligned} \tag{6.14b}$$

Notice that (6.14b) is the negation of (6.14a). Then, we use the theorem prover to prove that $T1$ does not entail $T2$ and $T2$ does not entail $T1$. Also, the theorem prover can prove that $T1$ entails $\neg T2$, so this case is classified as **C**.

Independence

Case 4 contains a pair of unrelated text. To classify this case, we build the logical forms in (6.15).

$$\begin{aligned} T1:\ & \text{思科}(n_0) \wedge property(n_0, n_6) \wedge \text{公司}(n_6) \wedge theme(n_6, e) \wedge \text{是}(e) \\ & \wedge \text{全球}(n_1) \wedge location(n_1, n_5) \wedge \text{最}(n_2) \wedge degree(n_2, n_5) \\ & \wedge \text{大}(n_5) \wedge property(n_5, n_7) \wedge \text{網路}(n_3) \wedge property(n_3, n_7) \\ & \wedge \text{供應}(n_4) \wedge property(n_4, n_7) \wedge \text{公司}(n_7) \wedge range(n_7, e) \end{aligned} \tag{6.15a}$$

$$\begin{aligned} T2:\ & \text{微軟}(n_{10}) \wedge theme(n_{10}, e) \wedge \text{是}(e) \wedge \text{全球}(n_1) \wedge property(n_1, n_7) \\ & \wedge \text{最}(n_2) \wedge degree(n_2, n_5) \wedge \text{大}(n_5) \wedge property(n_5, n_7) \\ & \wedge \text{軟體}(n_9) \wedge property(n_9, n_7) \wedge \text{公司}(n_7) \wedge range(n_7, e) \end{aligned} \tag{6.15b}$$

The theorem prover proves that $T1$ does not entail $T2$ and $T2$ does not entail $T1$. Also, it proves that $T1$ does not entail $\neg T2$, so this case is classified as **I**.

6.2.2 RITE Test Cases: Failed by Discussed Issues

In this section, we give some examples to show the causes of why our system failed to solve them. As expected, our system failed by some issues, and these issues are discussed in Chapter 5. These examples are in Table 6.4, and we will elaborate on the reasons for failure in the following paragraphs.

ID		Text	Label
Error in the CKIP Parser			
413	T1	人稱喬丹為籃球之神	R
	T2	邁克爾喬丹是籃球之神	
Exception in Semantic Constructor			
234	T1	靖國神社是供奉日本陣亡將士的地方	R
	T2	所有日本陣亡將士都供奉在靖國神社	
Exception in Knowledge Builder			
24	T1	希拉瑞爵士是首位成功攀上聖母峰的人	B
	T2	希拉瑞爵士即是有紀錄以來第一位登上聖母峰的人	
Exception in Knowledge Validator			
41	T1	小泉純一郎的長男是藝人小泉孝太郎	F
	T2	小泉純一郎的大兒子是小泉孝太郎	

Table 6.4: RITE Test Cases: Failed by Discussed Issues

Error in the CKIP Parser

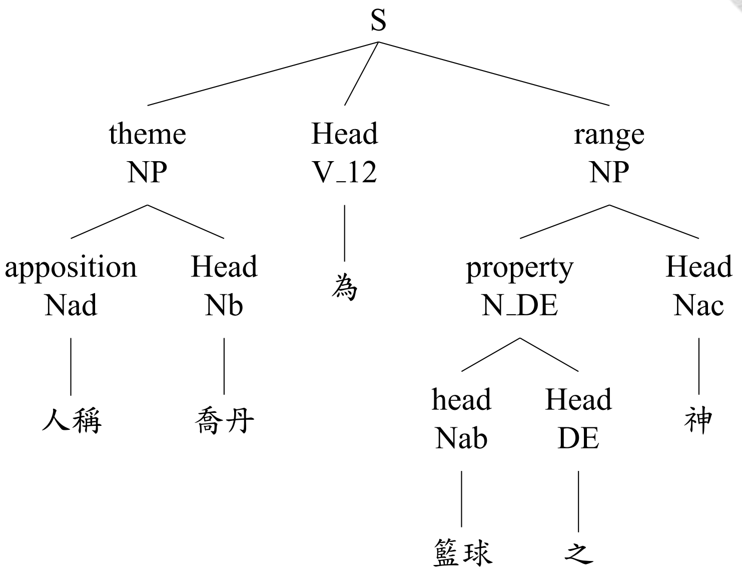
Case 413 has segmentation error in $T1$, and the resulting syntax tree is wrong, as illustrated in Figure 6.2a. Comparing to the correct syntax tree in Figure 6.2b. The wrong syntax tree has segmentation error in the words 人 (people) and 稱 (call), and the head word of the syntax tree is not 為 (as) but 稱 (call).

Exception in Semantic Constructor

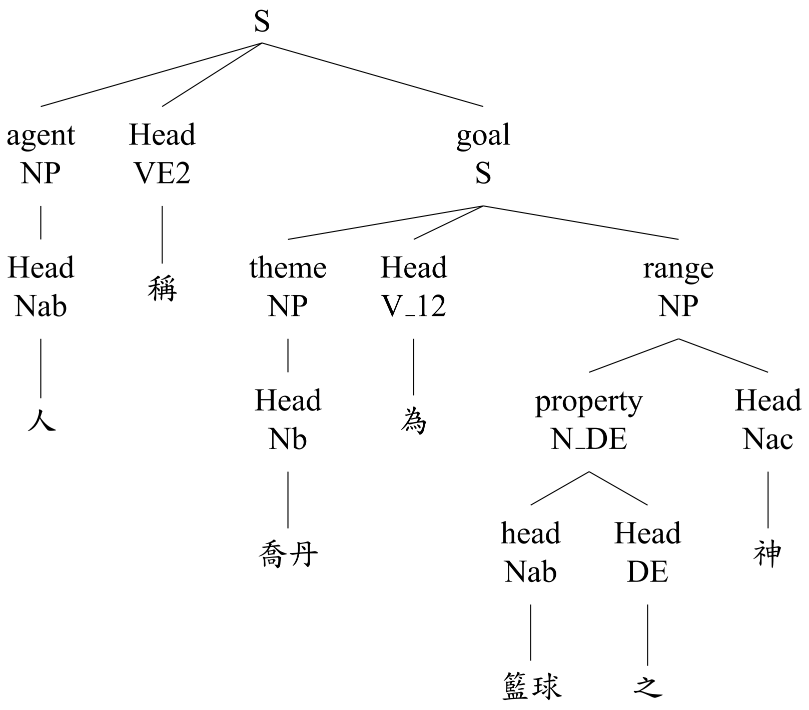
Case 234 contains the quantifier 所有 (all) in $T2$. However, as discussed in Section 5.2.1, we did not implement the algorithm to construct the logical form with quantifier.

Exception in Knowledge Builder

Case 24 has a noun phrase containing multiple words: 首位 (first) 成功 (success) 攀上 (climb on) 聖母峰 (Everest) 的 (DE) 人 (people) and 有 (has) 紀錄 (record) 以來 (since) 第一位 (first) 登上 (climb on) 聖母峰 (Everest) 的 (DE) 人 (people). However, as discussed in Section 5.3.1, our Knowledge Builder can't build the logical form of required knowledge of noun phrases with multiple words. Hence, this case can't be solved.



(a) Incorrect Syntax Tree



(b) Correct Syntax Tree

Figure 6.2: Syntax Tree of T1 in Case 413

Exception in Knowledge Validator

Case 41 needs extra knowledge to solve. The Knowledge Validator needs to validate the entailment between 長男 (elder son) 大兒子 (elder son), but it is not in the Chinese CWN. Moreover, the word 長男 (elder son) is not in the corpus of the Chinese LSA Website. Therefore, it can't calculate the semantic similarity between these two words.

6.2.3 RITE Test Cases: Failed by Other Issues

In the previous section, we discuss the causes why our system failed, and these causes are discussed in Chapter 5. However, it is possible that our system failed by other reasons we have not discussed. We analyze some of them, and give some examples of them in this section. These examples are shown in Table 6.5.

ID	Text		Label
Antonym			
341	T1	巴基斯坦的宿敵印度	C
	T2	巴基斯坦的好朋友印度	
Time Expression			
381	T1	桃莉誕生於一九九七年二月廿三日	C
	T2	早在 1996 年 7 月 5 日桃莉就誕生了	
Mathematics			
219	T1	甲烷燃燒時，與燒煤或燒油等燃料相比，釋放的二氧化碳可減少二分之一	B
	T2	燃燒煤或油釋放的二氧化碳是甲烷的兩倍。	
Sentence with Comma			
6	T1	1960 年 12 月馬漢德拉國王親政，1961 年 1 月宣布禁止一切政黨活動	F
	T2	1960 年馬漢德拉國王恢復君主專制	
Idiom			
73	T1	安南把全球關注焦點集中在貧窮、違反人權、非洲的衝突和愛滋病蔓延等	C
	T2	安南對於貧窮、違反人權、非洲的衝突和愛滋病蔓延等問題漠不關心	

Table 6.5: RITE Test Cases: Failed by Other Issues

Antonym

Case 341 requires extra knowledge to solve. The theorem prover can prove that $T1$ does not entail $T2$ and $T2$ does not entail $T1$. Since the label of this case is **C**, we need to prove that $T1$ entails $\neg T2$. We need to have the extra knowledge that the word 宿敵 (enemy) is the antonym of the word 好朋友 (friend). Since we only consider hypernym and synonym in our Knowledge Builder, this situation is outside the scope of this work.

Time Expression

Case 381 contains two time expressions in $T1$ and $T2$. They are "一九九七年二月廿三日" (2.23.1997) and "1996 年 7 月 5 日" (7.5.1996). The formats of time expressions are different. If we want to solve this problem, we need to convert them into the same format. Also, the semantic representation should be able to express the time expressions. However, it is outside the scope of this work.

Mathematics

Case 219 contains two numerical terms " 二分之一"(half,$\frac{1}{2}$) and " 兩倍"(twice,2) in $T1$ and $T2$. Since the word half($\frac{1}{2}$) is of a fractional number, and its reciprocal is 2. Solving this problem requires the reciprocal operation.

Sentence with Comma

Case 6 contains a sentence with two segments divided by a comma "，" in $T1$. The subject in the sub-sentence in the right-hand side of the comma is implicitly the same as the subject in the left-hand side of the comma. The resulting semantic representation should have the same subject, too. However, we did not handle this situation.

Idiom

Case 73 contains the idiom 漠不關心 (indifference) in $T2$. To solve this, we need to know that the meaning of the idiom 漠不關心 is contrary to the phrase 集中在 (focus on).

Chapter 7

Conclusion

During the experiment, we demonstrate how to construct the logical forms and how to solve the RTE problems. After this experiment, we know that some problems can be solved by our system, while others can't be solved. In this section, we will make a brief summary of our work, and compare our work with other previous works. We show the pros and cons of our system from the two aspects: Chinese computational semantics and Chinese RTE.

7.1 About Chinese Computational Semantics

In this work, we present a new technique to convert Chinese raw sentence into logical form. First, the raw sentence is parsed by the CKIP Chinese Parser. Then, the resulting syntax tree is converted into Neo-Davidsonian semantic representation. We use a straightforward algorithm with five operations to build the logical form. For some special words, such as DE, prepositions, coordination and negation, we slightly adjust those operations to construct the semantic representations related to these words. However, there are still other semantic expressions that we have not implement in the Semantic Constructor, such as quantifier or anaphora, which require more advanced theory in formal semantics to express their semantics. We compare our work with other previous works.

Comparing to the previous work Chen and Wu [36], first, they adopt a rule-based parser and can only parse limited sentences in the Sinica Balanced Corpus [37]. Our work can build logical form from any grammatically correct sentence in Chinese. We achieve a wider coverage semantic constructing algorithm comparing to theirs. Second, they implement the generalized quantifier theory [38] and their work can express the semantics of quantifier. However, our work is incapable of building the semantics of quantifier. The expressiveness of logical form in our work is weaker than theirs. Third, their work has no evaluation, but we present a method to evaluate the semantic representation by an RTE experiment.

Comparing to the previous work Zhang and Zhang [34], first, their work can also converts raw sentence into logical form, but they adopt the Davidson semantic representation without thematic roles. The semantic representation in our work has thematic roles. Hence, we have the richer semantic information in logical form. Second, their work extracts external knowledge from knowledge resources, HowNet [35]. Our work uses an algorithm to build the required knowledge in logical form according to input text, and validate the knowledge by external knowledge resources. By this method, we can avoid to build the logical form of unnecessary knowledge, but some required knowledge of logical forms are outside the capability of our Knowledge Builder. Third, they use the task of question answering to evaluate their system, and we use the RTE task. Both of the methods are effective to demonstrate the capabilities of the systems.

7.2 About Chinese RTE

We present a formal logic method to solve the tasks of Chinese RTE. First, we present an algorithm to solve the Chinese RTE problems by converting the input text into logical form, and construct the logical form of required knowledge. The required knowledge is constructed according to the input logical forms, and can be validated or invalidated by external knowledge resources. Then, we use the theorem proving algorithm to prove that whether the entailment is true or not. To the best of our knowledge, up to the present, our work is the only work that uses formal logic method to solve the Chinese RTE tasks.

We compare the pros and cons of our work with other previous Chinese RTE works adopting machine learning techniques. There are two limitations in our system. First, our work can only solve the RTE cases with simple sentences successfully, and can only solve relatively few cases in the RITE competition. Due to the errors in the CKIP Parser, Semantic Construction, Knowledge Builder and Validator, these errors can be propagated through each stage, resulting in a huge limitation of our work to solve the test cases correctly. In contrast, machine learning technique is not susceptible to the errors in any stage. Second, to build semantic representation and the logical form of required knowledge, our system need to build them case by case, since there is no general algorithm to solve all the cases without any exception. On the contrary, with machine learning technique, implementing a simple model is enough to solve all the cases. However, the strength of our system is that we do not need to use any training data to train our system. We can directly calculate the truth of whether T entails H. By comparison, machine learning techniques require huge amount of training data to achieve the accuracy of classifying the truth. Most important of all, our work gives a perspective of solving the RTE problem with the spirit of natural language understanding. We use the semantic representation and external knowledge to make the machine understand the meaning of these test cases.

Chapter 8

Future Works

This is the first time that we do research on the topic of computational semantics and natural language understanding. Our system can serve as a framework for our further research on this topic. In the future, we will proceed to improve our system and we hope that our system can be used in the real-world application. We point out some perspectives for future research.

8.1 Chinese Word Segmentation and Parsing

The precision of the Chinese word segmentation and parsing are crucial to our work to successfully solve the RTE task. We will search for more accurate Chinese word segmentator and parser. Also, we will try to improve the precision of word segmentation and parsing by some plausible methods.

8.2 Semantic Construction

In the future work, we need to enhance the expressiveness of our semantic representation. We can use generalized quantifier theory [38] to construct the semantics of quantifier. Also, we can use discourse representation theory [62] to construct the semantics with anaphora. With this theory, it is possible to solve the RTE case with numerous sentences or even an entire article. Moreover, we can construct the semantics about temporal expression and mathematical expression. With the enhanced expressiveness, our system can handle more cases in the RITE test case.

8.3 Reasoning Algorithm

In this research, we employ the theorem proving technique as reasoning algorithm. However, for the knowledge extracted from distributional semantics, we can't guarantee that the knowledge is absolutely valid. If we use abductive reasoning algorithm [49], we can do reasoning with this kind of ambiguous knowledge.

8.4 Knowledge Resources and Knowledge Construction

In this research, we use a simple algorithm to construct the logical form of required knowledge from logical form of input text. However, this method can't construct all the required knowledge in logical form; hence, more sophisticated algorithm is required. On the other hand, we can use more diverse knowledge resources. For example, we can use ontology as knowledge resources for world knowledge. We can also obtain knowledge from the web corpus. It is feasible to automatically construct knowledge resources from web data by using machine learning algorithm, such as clustering or distributional semantics. Also, it is possible to obtain the required knowledge from search engine.

8.5 Real-World Application

The most important of all, our research should not be limited to academic world. In the future, our system should be able to integrate into some real-world applications. Since RTE can be applied to many real-world applications, such as question-answering system, search engine, and document summarizing system.

Bibliography

[1] James Allen. *Natural Language Understanding*. Benjamin Cummings, Menlo Park, CA, 1995.

[2] Ekaterina Ovchinnikova. *Integration of World Knowledge for Natural Language Understanding.* Atlantis Thinking Machines. Atlantis Press, 2012.

[3] Donald Davidson. The individuation of events. In N. Resher, editor, *Essays in Honor of Carl G. Hempel*, page 216 – 234. Springer, 1969.

[4] David R. Dowty. On the semantic content of the notion of 'thematic role'. In Raymond Turner Gennaro Chierchia, Barbara H. Partee, editor, *Properties, Types and Meaning*, pages 69–129. 1989.

[5] Terence Parsons. *Events in the Semantics of English: A Study in Subatomic Semantics*. MIT Press, 1990.

[6] Patrick Blackburn and Johan Bos. *Representation and Inference for Natural Language. A First Course in Computational Semantics*. CSLI, 2005.

[7] Ido Dagan, Oren Glickman, and Bernardo Magnini. The pascal recognizing textual entailment challenge. In *Proceedings of the PASCAL Challenges Workshop on Recognizing Textual Entailment*, 2005.

[8] Marta Tatu and Dan Moldovan. A logic-based semantic approach to recognizing textual entailment. In *Proceedings of the COLING/ACL on Main Conference Poster Sessions*, COLING-ACL '06, pages 819–826, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics.

[9] S. R. Safavian and D. Landgrebe. A survey of decision tree classifier methodology. *Systems, Man and Cybernetics, IEEE Transactions on*, 21(3):660–674, 1991.

[10] Corinna Cortes and Vladimir Vapnik. Support-vector networks. In *Machine Learning*, pages 273–297, 1995.

[11] Steven Bird, Ewan Klein, and Edward Loper. Supervised classification. In *Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit*, pages 221–231. O'Reilly, Beijing, 2009.

[12] Christiane Fellbaum. *WordNet: An Electronic Lexical Database*. MIT Press, 1998.

[13] Karin Kipper, Anna Korhonen, Neville Ryant, and Martha Palmer. A large-scale classification of english verbs. *Language Resources and Evaluation*, 2007.

[14] Thomas R. Gruber. A translation approach to portable ontology specifications. *Knowl. Acquis.*, 5(2), June 1993.

[15] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, New York, NY, USA, 2003.

[16] Herbert Rubenstein and John B. Goodenough. Contextual correlates of synonymy. *Commun. ACM*, 8(10):627–633, 1965.

[17] T.K. Landauer, P.W. Foltz, and D. Laham. An introduction to latent semantic analysis. *Discourse processes*, 25:259–284, 1998.

[18] Kevin Lund, Curt Burgess, and Ruth A. Atchley. Semantic and associative priming in high-dimensional semantic space. In *Proceedings of the 17th Annual Conference of the Cognitive Science Society*, pages 660–665. Hillsdale, NJ: Erlbaum, 1995.

[19] Johan Bos. A survey of computational semantics: Representation, inference and knowledge in wide-coverage text understanding. *Language and Linguistics Compass*, 2011.

[20] Richard Montague. The proper treatment of quantification in ordinary english. In Richmond H. Thomason, editor, *Formal Philosophy*, page 247 – 270. 1973.

[21] Carl Pollard Ann Copestake, Dan Flickinger and Ivan A. Sag. Minimal recursion semantics: An introduction. 3(2-3):281 – 332, 2005.

[22] Josef van Genabith, Anette Frank, and Dick Crouch. Glue, underspecification and translation. page 265 – 279, 1999.

[23] Christof Monz and Maarten de Rijke. Light-weight entailment checking for computational semantics. 2001.

[24] Allan Ramsay and Helen Seville. Models and discourse models. 1(2):167 – 181, 2000.

[25] Dan Moldovan and Vasile Rus. Explaining answers with extended wordnet. In *ACL*, 2001.

[26] Collin F. Baker, Charles J. Fillmore, and John B. Lowe. The berkeley framenet project. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics - Volume 1*, ACL '98, pages 86–90, Stroudsburg, PA, USA, 1998. Association for Computational Linguistics.

[27] Martha Palmer, Paul Kingsbury, and Daniel Gildea. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31, 2005.

[28] Stephen Clark and James R. Curra. Wide-coverage efficient statistical parsing with ccg and log-linear models. 33(4):493–552, 2007.

[29] Mark Steedman. *The Syntactic Process*. MIT Press, 2000.

[30] Dick Crouch and Tracy Holloway King. Semantics via f-structure rewriting. In *Proceedings of the LFG06 Conference*, 2006.

[31] Ronald M. Kaplan and Joan Bresnan. Lexical-functional grammar: A formal system for grammatical representation, 1995.

[32] Ann Copestake and Dan Flickinger. An open source grammar development environment and broad-coverage english grammar using hpsg. In *Proceedings of LREC 2000*, pages 591–600, 2000.

[33] Carl Pollard and Ivan Sag. *Head-Driven Phrase Structure Grammar*. Studies in Contemporary Linguistics. University of Chicago Press, 1994.

[34] YI Zhang and Dong Mo Zhang. Enabling answer validation by logic form reasoning in chinese question answering. In *Proceeding of 2003 International Conference on Natural Language Processing and Knowledge Engineering*, pages 275– 280, 2003.

[35] Dong Zhendong and Dong Qiang. HowNet, 1999, cited May 2014. URL http://www.keenage.com/.

[36] Szu-Hua Chen and Jiun-Shiung Wu. Toward a computational semantic grammar for mandarin chinese: A sinica corpus-based study. Master's thesis, Graduate Institute of Linguistics, National Chung Cheng University, 2013.

[37] Li-Ping Chang Chen Keh-Jiann, Chu-Ren Huang and Hui-Li Hsu. Sinica corpus: Design methodology for balanced corpra. In *PACLIC*, pages 167–176, 1996.

[38] Jon Barwise and Robin Cooper. Generalized quantifiers and natural language. *Linguistics and Philosophy*, 4(2):159–219, 1981.

[39] The Fracas Consortium, Robin Cooper, Dick Crouch, Jan Van Eijck, Chris Fox, Josef Van Genabith, Jan Jaspars, Hans Kamp, David Milward, Manfred Pinkal, Massimo Poesio, Steve Pulman, Ted Briscoe, Holger Maier, and Karsten Konrad. Using the framework, 1996.

[40] Ion Androutsopoulos and Prodromos Malakasiotis. A survey of paraphrasing and textual entailment methods. *CoRR*, abs/0912.3747, 2009.

[41] Andreas Wotzlaw and Ravi Coote. A logic-based approach for recognizing textual entailment supported by ontological background knowledge. *CoRR*, abs/1310.4938, 2013.

[42] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: A core of semantic knowledge. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, pages 697–706, New York, NY, USA, 2007. ACM.

[43] Cynthia Matuszek, John Cabral, Michael Witbrock, and John Deoliveira. An introduction to the syntax and content of cyc. In *Proceedings of the 2006 AAAI Spring Symposium on Formalizing and Compiling Background Knowledge and Its Applications to Knowledge Representation and Question Answering*, pages 44–49, 2006.

[44] Fabio Massimo Zanzotto, Marco Pennacchiotti, and Alessandro Moschitti. A machine learning approach to textual entailment recognition. *Nat. Lang. Eng.*, 15(4): 551–582, 2009.

[45] Hen-Hsen Huang, Kai-Chun Chang, and Hsin-Hsi Chen. Modeling human inference process for textual entailment recognition. In *ACL*, pages 446–450. The Association for Computer Linguistics, 2013.

[46] Johan Bos and Katja Markert. Recognizing textual entailment with logical inference. In *EMNLP-05*, pages 628–635, 2005.

[47] Rajat Raina, Andrew Y. Ng, and Christopher D. Manning. Robust textual inference via learning and abductive reasoning. In *Proc. of AAAI 2005*, pages 1099–1105, 2005.

[48] Dan Klein and Christopher D. Manning. Accurate unlexicalized parsing. In *IN PROCEEDINGS OF THE 41ST ANNUAL MEETING OF THE ASSOCIATION FOR COMPUTATIONAL LINGUISTICS*, pages 423–430, 2003.

[49] J.R. Hobbs, M.E. Stickel, D.E. Appelt, and P. Martin. Interpretation as abduction. *Artificial Intelligence*, 63:69–142, 1993.

[50] Hideki Shima, Hiroshi Kanayama, Cheng-Wei Lee, Chuan-Jie Lin, Teruko Mitamura, Yusuke Miyao, Shuming Shi, and Koichi Takeda. Overview of ntcir-9 rite: Recognizing inference in text, 2011.

[51] Yotaro Watanabe, Yusuke Miyao, Junta Mizuno, Tomohide Shibata, Hiroshi Kanayama, Cheng-Wei Lee, Chuan-Jie Lin, Shuming Shi, Teruko Mitamura, Noriko Kando, Hideki Shima, and Kohichi Takeda. Overview of the recognizing inference in text (rite-2) at ntcir-10, 2013.

[52] Hen-Hsen Huang, Kai-Chun Chang, James M.C. Haver II, and Hsin-Hsi Chen. Ntu textual entailment system for ntcir 9 rite task, 2011.

[53] Shan-Shun Yang, Shih-Hung Wu, Liang-Pu Chen, Hung-Sheng Chiu, and Ren-Dar Yang. Entailment analysis for improving chinese recognizing textual entailment system. In *ROCLING*, 2013.

[54] Chinese Knowledge Information Processing Group. CKIP Chinese Parser, cited May 2014. URL http://ckip.iis.sinica.edu.tw/CKIP/parser.htm.

[55] Chinese Knowledge Information Processing Group. Technical report no. 93-05：中文詞類分析 (三版). Technical report, Institute of Information Science, Academia Sinica, 1993.

[56] Chinese Knowledge Information Processing Group. Technical report no. 13-01：句結構樹中的語意角色. Technical report, Institute of Information Science, Academia Sinica, 2013.

[57] Chu-Ren Huang and Shu-Kai Hsieh. Infrastructure for Cross-lingual KnowledgeRepresentation —Towards Multilingualism in Linguistic Studies. Taiwan NSC-granted Research Project (NSC 96-2411-H-003-061-MY3) , 2010, cited May 2014. URL http://lope.linguistics.ntu.edu.tw/cwn/.

[58] Hua wei Ke, Ming lei Chen, and Xue cheng Wang. Chinese latent semantic analysis website, 2009, cited 2014. URL http://www.lsa.url.tw/modules/lsa/.

[59] Steven Bird. Nltk: The natural language toolkit. In *Proceedings of the COLING/ACL on Interactive presentation sessions*, COLING-ACL '06, pages 69–72, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics.

[60] W. McCune. Prover9 and mace4. http://www.cs.unm.edu/~mccune/prover9/, 2005–2010.

[61] Roberto Navigli. Word sense disambiguation: a survey. *ACM COMPUTING SURVEYS*, 41(2):1–69, 2009.

[62] Hans Kamp. Discourse representation theory. In J. Verschueren, J.-O. Östman, and J. Blommaert, editors, *Handbook of Pragmatics*, pages 253–257. Benjamins, 1995.