A GENERAL PURPOSE SEMANTIC PARSER USING FRAMENET AND WORDNET

Lei Shi, B.E.

Thesis Prepared for the Degree of

MASTER OF SCIENCE

UNIVERSITY OF NORTH TEXAS

May 2004

APPROVED:

Rada Mihalcea, Major Professor
Robert Brazile, Program Coordinator in the Department of Computer Science
Krishna Kavi, Chair of the Department of Computer Science
Oscar Garcia, Dean of the College of Engineering
Sandra L. Terrell, Interim Dean of the Robert B. Toulouse School of Graduate Studies

Shi, Lei, A general purpose semantic parser using FrameNet and WordNet®, Master of Science (Computer Science), May 2004, 50 pp., 3 tables, 5 figures, references, 14 titles.

Syntactic parsing is one of the best understood language processing applications. Since language and grammar have been formally defined, it is easy for computers to parse the syntactic structure of natural language text. Does meaning have structure as well? If it has, how can we analyze the structure? Previous systems rely on a one-to-one correspondence between syntactic rules and semantic rules. But such systems can only be applied to limited fragments of English. In this thesis, we propose a general-purpose shallow semantic parser which utilizes a semantic network (WordNet), and a frame dataset (FrameNet). Semantic relations recognized by the parser are based on how human beings represent knowledge of the world. Parsing semantic structure allows semantic units and constituents to be accessed and processed in a more meaningful way than syntactic parsing, moving the automation of understanding natural language text to a higher level.

ACKNOWLEDGEMENTS

I would like to acknowledge all those who helped make this thesis a reality. In particular, I would like to acknowledge Dr. Rada Mihalcea for her invaluable advice, support and guidance, which are very important to the thesis. I would like to acknowledge Dr. Paul Tarau and other members in NLP group for sharing their knowledge and interesting ideas in various meetings and conversations, Dr. Kathy Swigger for teaching me "Natural Language Processing" and "Artificial Intelligence". I would also like to acknowledge my parents and friends for their emotional support and concern.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Noam Chomsky introduced formal definitions about language and its grammar so that the syntactic structure of languages can be analyzed by computer programs. Syntax is the study of language from a purely formal point of view, usually with no attention to meaning. If we were to talk about a natural language, we could go on and say a great deal about the language from this purely formal point of view. Some linguists in our century imply that this is all there is to say, that the only important thing about language is the network of formal relationships and contrasts that exist in the language. In early days, some of the most influential linguists such as Leonard Bloomfield and Zellig Harris had this idea. Of course, they recognized that words have meanings, but they thought the study of meaning could not be done in a precise and scientific way. In this respect, they agreed with many philosophers and logicians who said that natural languages are so vague and ambiguous that they cannot be described in the same way as artificial languages, such as PASCAL, can he described.

Richard Montague was one philosopher who did not agree with this view. In his paper written in late 60's and 70's, Montague claimed that natural language could be treated in just the same way as the formal artificial languages of logician. Montague's theory is TRUTH-CONDITIONAL and MODEL-THEORETIC. Truth-condition means that the meaning of a sentence is the set of necessary and sufficient conditions for the sentence to be TRUE. Model-theoretic means that the theory uses a formal mathematical model of the world in order to set up relationships between linguistic elements and their meanings. Montague's system contains a set of syntactic rules and a set of semantic rules, and the two are in one-to-one correspondence–each time a particular syntactic rule applies, so does the corresponding semantic rule.

Although Montague semantics has much to recommend it and in contemporary linguistic theory there has been a continuation of this view in work inspired largely by Montague, its truth-conditional semantics is not very useful in AI systems. We are not so interested in whether a state of affair is or could be true in some possible world,

but rather in the state of affair itself; thus AI uses KNOWLEDGE-BASE SEMANTICS in which semantic objects tend to be symbols or expressions in a declarative or procedural knowledge-representation system. Moreover, truth-conditional semantics only deals with declarative sentences, while a practical NLP system should be able to handle commands, queries etc. Another problem, which may be the biggest one, in Montague's semantic interpretation system is that it relies on a one-to-one correspondence between syntactic rules and semantic rules. In practice, syntactic rules may have different semantic implications in different situations, so that Montague's system only applies to a small fragment of English.

This thesis presents a semantic parser which employs formal procedures to provide a formal semantic representation of sentence meaning. The semantic parser introduced in this thesis is different from the truth-conditional logical form in two respects: First, the semantic parser focuses more on the structure of sentence meaning rather than representing the meaning itself. Instead of finding the truth conditions in a mathematical possible world and obtaining logical inference, the semantic parser analyzes how smaller semantic units (such as semantics of words and phrases) can compose a bigger one (such as that of phrases and sentences) through various semantic relations, which is explicitly specified. In the semantic parser, the smallest semantic unit—the word is not interpreted, while the meaning of the whole sentence is interpreted by showing how the sentence semantics is composed with these units. This makes the acquisition of particular semantic information easy and allows more flexible processing on semantics in addition to logical inference. In the mean time, it is easy to translate the semantic parse tree into a truth-conditional logical form. Second, syntax is no longer the only tool for identifying semantic relations and the ontology based semantic parser does not rely on a one-to-one correspondence between syntax and semantics. Natural language is used by humans to represent the world, and a natural language understanding system in most cases needs to possess human ontology in order to produce reasonable output. To this end, semantic relations are defined according to how human beings represent their knowledge about the world. Ontology plays an important role in semantic analysis.

The theory and implementation of semantic parsing is elaborated in chapter 3.

Before that, chapter 2 introduces the related background theory, the data and tools used in developing the parser. Chapter 4 deals with possible applications of the semantic parser in various fields of NLP and future works.

CHAPTER 2

BACKGROUND

This chapter introduces some background theories and technique used in the semantic parser including syntactic parsing with features, parser, logic programming and some database such as WordNet, FrameNet.

2.1 Parsing and Context Free Grammar

Parsing usually refers to syntactic parsing. It is the method of analyzing a sentence to determine its syntactic structure. The syntactic structure tells us how words in a sentence are related to each other, how they are grouped to form phrases and how phrases are grouped to form the sentence. In addition, the structure may indicate the types of relationships between words and phrases and provide other information about the particular sentence structure for further processing.

There are two things needed for parsing. One is the Grammar, and the other one is the Parser. The grammar is a formal specification of legal syntactic structures allowable in the language. The parser is a computer program that analyzes a sentence to determine its syntactic structures based on the grammar. The grammar tells the parser how it parses a sentence which belongs to the language the grammar defines, and the output of the parser is the syntactic structure of the sentence, which is usually like a tree.

People usually think of language and grammar as our everyday English and English grammar. But from a computational point of view, the notion of language and grammar is far beyond that. So, we shall introduce some important definitions of terms and the formal definition of language and grammar in computational linguistics, in which, the context free grammar and context free language are given more attention since they are powerful enough to describe most of the structure in natural language. Later, an example of a small context free grammar and parsing algorithm is

provided, and the parser will show the structure of a sentence that can be recognized by the grammar.

Alphabets: An alphabet is a finite, nonempty set of symbols. Usually, we use S for an alphabet. An alphabet is a set, and the symbols can be any symbol. For instance, the $\sum$ can be $\{b, o, x, y\}$, or $\{1, 2, 3, 4, 5\}$ or $\{a, B, 3, \$\}$.

Strings: A string is a finite sequence of symbols chosen from some alphabet. For the example above $\sum = b, o, x, y$, the string "*box*" is a valid string for the alphabet $\sum$, the string "*by*" is also a valid string for the alphabet $\sum$, but the string "*xok*" is not a valid string for alphabet $\sum$ because k is not in the alphabet.

Languages: A set of strings all of which are chosen from some $\sum$, where $\sum$ is a particular alphabet, is called a language, and we can call it a language over alphabet $\sum$. For $\sum = a, b$,the set of strings "*a*", "*b*", "*ab*", "*ba*" establish a language L over $\sum$. Notice that a language over $\sum$ does not need to include strings with all the symbols of $\sum$. So the language $L =$ "*a*" is also a language over $\sum = a, b$.

Corresponding to the natural language of English, we can think of the alphabet of the language as English letters $a - z$ and $A - Z$, which is the alphabet of English, think of strings as English words, phrases and sentences because every English word, phrase and sentence come from the alphabet. The English language is the set of such strings. So we can say the strings "good", "nice person" and "I am the superman" belong to English language. In the other hand, the string "I is the superman" is not part of English, because it violates the English Grammar. As we said before, the language need not include every string over its alphabet, so the function of Grammar is to define what strings over the alphabet constitute the language. So in order to define a language, we need to define an alphabet and a grammar.

According to Chomsky's theory, the language is classified into three sets or categories: Regular Expression, Context Free Language and Turing Machine, the latter, the more powerful, which means the former category is the subset of the latter category and the Turing Machine is the most general language. Correspondingly, the Context Free Grammar (CFG) defines a Context Free Language. Here, we will only talk about CFG because it defines most of the structure of natural language and our parser will use CFG to determine the syntactic structure of the language.

A Context Free Grammar is a set of recursive rewriting rules(or productions) used to generate patterns of strings. It is composed of:

A set of *terminal symbols*, which are the characters of the alphabet that appear in the string generated by the grammar.

A set of *nonterminal symbols*, which are the placeholders for patterns of terminal symbols that can be generated by the nonterminal symbols.

A set of *productions*, which are the rules for replacing(rewriting) nonterminal symbols on the left side of the production with other terminal or nonterminal symbols on the right side of the production.

A *start symbol*, which is a nonterminal symbol that represents the strings generated by the grammar.

Below we take a simple CFG as an example and show what language (the set of strings) can be recognized by the grammar.

$S \rightarrow$ NP VP
$NP \rightarrow$ N
$NP \rightarrow$ PRON
$VP \rightarrow$ V N
$N \rightarrow$ dog
$N \rightarrow$ cat
$V \rightarrow$ chase
$PRON \rightarrow$ you

Here words of capital letters are nonterminal symbols and others terminal symbols. We can say S is the start symbol. The alphabet is the set of letters used by those terminal symbols or superset of it. Now we have defined a language. Given a string "dog chase cat", based on the grammar, we can testify if it belongs to the language. Let's apply the rewriting rules one by one from the start symbol and see if the string can be reached.

From the start symbol S, since S is a nonterminal, we apply the rule $S \rightarrow$ NP VP.

NP is a nonterminal, and we can rewrite it as N or PRON. We use N first and it can be rewritten to a terminal "dog". Since NP in $S \rightarrow$ NP VP has found a terminal replacement of "dog", we track back and begin to find replacement for VP. VP is rewritten to V and N. V can be replaced by the terminal "chase" and N can be replaced by "cat", so VP can be rewritten as "chase cat". Now the start nonterminal symbol S can be replaced by terminal "dog chase cat", which is exactly the same as the string we are testifying. In this case, we can say the string is recognized by the language we just defined, although it is grammatically wrong for the language of English.

If we think in a reverse way, the string can be parsed as shown in figure 2.1:

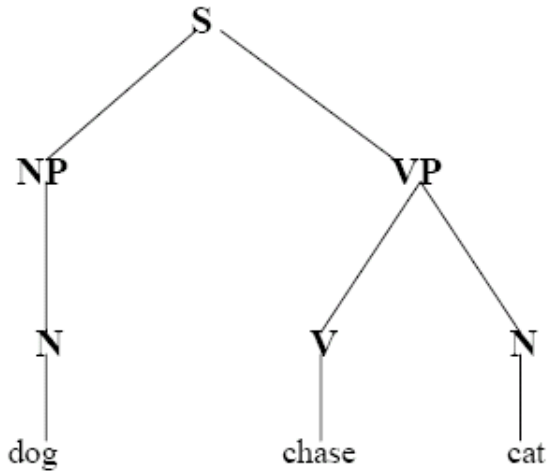

Figure 2.1: Syntax tree of "dog chase cat"

A parser is the computer program that can determine the structure of a sentence like the diagram above based on the grammar that defines it. My former description of how to find the string is recognized by the grammar can be slightly modified to be a simple parsing algorithm. We start from the nonterminal start symbol S and apply every rule that can replace the nonterminal symbol with other nonterminal symbols or terminal symbols. In turn, new nonterminal symbols are further replaced by their rewriting rules from left to right, until all symbols are terminal symbols of the string. We keep the information of rules used and nonterminal symbols along the way we

reach the string, and the parse tree then can be generated. So this simple algorithm is actually a depth first top down search algorithm.

An interesting phenomenon of a context free language is ambiguity. Ambiguity means the same string can be parsed in two or more different ways according to the grammar. The CFG below defines a context free language

$S \rightarrow$ NP VP
$S \rightarrow$ NP VP NP
$VP \rightarrow$ VT N
$VP \rightarrow$ VT
$NP \rightarrow$ N
$NP \rightarrow$ NAME
$NAME \rightarrow$ stone
$N \rightarrow$ superman
$VT \rightarrow$ is

We can see that the string "stone is superman" can be correctly recognized by the grammar, but the problem is the string has two different parse trees as below.

Parse tree A: see figure 2.2

Parse tree B: see figure 2.3

Not only in this small grammar we just define, ambiguity is also very common in natural language. A very typical example is "I saw a man with a telescope". In this case, we really cannot identify "with a telescope" should be attached to "I" or "a man", without context. Ambiguity is a very common and hard problem in syntactic analysis of natural language. Since sometime syntactic structure determines the semantic structure, the semantic parser may also suffer from this problem. One of the major solutions is to let the computer learn the probability of a specific parsing from a large collection of corpus. Since it's beyond the topic of the thesis, I won't elaborate on that.

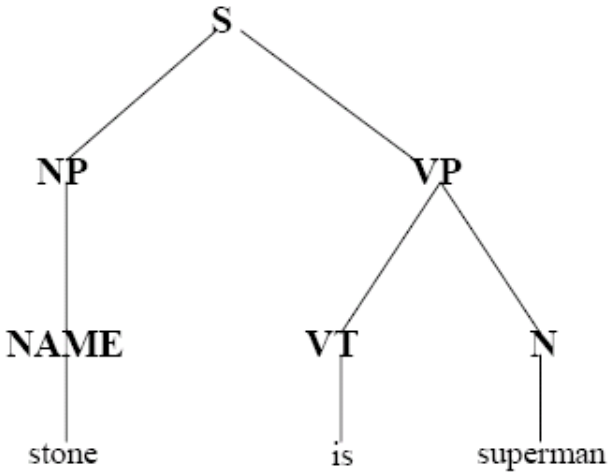

Figure 2.2: Syntax tree 1 of “stone is superman”

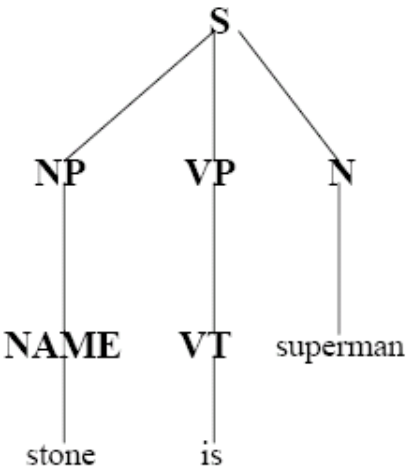

Figure 2.3: Syntax tree 2 of “stone is superman”

2.2 Logic Programming and Prolog

Logic programming is a declarative, relational style of programming based on first-order logic. Logic programs consist of logical formulas and computation is the process of deduction or proof construction. What makes logic programming fundamentally different from most other programming languages is largely the fact that logic is much older than digital computers and not restricted to the view of computation associated with the Von Neumann machine. The main difference between conventional programming and logic programming is the *declarative* nature of logic. A program written in, say, PASCAL, in general cannot be understood without taking operational considerations into account. That is to say, we cannot understand a piece of PASCAL code without knowing how it is executed. In contrast, logic does not have inherent concept of execution and logic programs can be understood without any evaluation of execution in mind. The construction of a computer program can be divided into two phases, which are usually intertwined — the formulation of the actual problem (what the problem is) and the description of how to solve the problem. Together they form an algorithm. This idea is the kernel of logic programming. The logic provides the description of the problem and the theorem prover, more formally SLD-resolution, performs the execution of the description. The theorem prover is a deductive prover, which is built in the logic programming languages and the logic programmer's job is to declaratively describe the problem with logic.

Prolog is the original logic programming language. It was invented by Alain Colmerauer and Phillipe Roussel at the University of Aix-Marseille in 1971. It was designed originally for natural language processing but has become one of the most widely used languages for artificial intelligence.

To write a prolog program, there are three basic constructs: facts, rules and queries. A collection of facts and rules that describe a problem is called a knowledge base and that is what Prolog programming is all about. Prolog has an inherent theorem prover based on matching and backtracking, which can deduce new facts from the knowledge base. The interaction between users and the program is through dialog. By posing queries, users can get answers or solutions to the problem. Below

is a small example of Prolog code:

```
father(tom,jerry).
mother(linda,jerry).
parent(X,Y):- father(X,Y).
parent(X,Y):- mother(X,Y).
```

The first two clauses are facts, which mean tom is jerry's father and linda is jerry's mother. While the last two clauses are rules, which define if someone, denoted by variable X is the father (or mother) of someone else, denoted by variable Y, then X is Y's parent. The problem is now described. If a user poses a query parent(linda,Who), which means whose parent is linda. Let's see how prolog's theorem prover solves the problem with its matching and backtracking mechanism. Prolog will first search for the relation parent starting from the first clause. When it reaches the first rule, variable X is matched to linda, then it is going to find a match for variable Y. The first rule tells Y can be matched to something if X is the father of Y. Then prolog will start the search again for relation "father(linda,Y)", but there is no such fact or rule that could derive from, then this attempt of matching fails. There is another rule for matching parent(linda,Who), which is to find whose mother is linda. Of course by searching from the beginning again, prolog finds a fact mother(linda,jerry), then Who is matched to jerry. Then the prolog has found an answer and output "Who = jerry". All these are handled by prolog. The trace of the proof shows that the prolog proof strategy is a depth first top down search strategy.

The nature of declarative description of problems of logic programming is particularly useful for natural language processing. If each word in a sentence is numbered by its position, the production rule such as $VP \rightarrow$ V N can be reformulated as an axiom that says, "There is a VP between P1 and P3, if there is a position P2 such that there is an NP between P1 and P2 and a VP between P2 and P3."

```
vp(P1,P3):- v(P1,P2),n(P2,P3).
```

The lexicon may be defined as follows:

```
isn(dog).
n(P1,P2):- word(Word,P1,P2),isn(Word).
```

So the context free grammar defined in Figure 2-1 can be translated into following rules as:

```
s(P1,P3):- np(P1,P2),vp(P2,P3).
np(P1,P2):- n(P1,P2).
np(P1,P2):- pron(P1,P2).
vp(P1,P3):- v(P1,P2),n(P2,P3).
n(P1,P2):- word(Word,P1,P2),isn(Word).
v(P1,P2):- word(Word,P1,P2),isv(Word).
pron(P1,P2):- word(Word,P1,P2),ispron(Word).
```

The string "dog chase cat" can be translated into:

```
word(dog,1,2).
word(chase,2,3).
word(cat,3,4).
```

If a query s(1,4) is asked to prolog, its depth first top down search proof engine will recognize the string the same way the simple top down parser does and output "yes".

With the powerful logic programming language Prolog, the problem of parsing sentences in a language is reduced to the problem of describing the language which can be easily coded by prolog in a declarative manner.

2.3 Bottom up Chart Parser with Features

Context Free Grammars provide the basis for most of the computational parsing mechanism developed to date. But in practice, CFG would be very inconvenient for capturing natural language. For example, a books is not a correct English noun phrase, because the number of the determiner a does not agree with that of the noun books. In order to capture this phenomenon of English, the Context Free Grammar has to use to two different nonterminal symbols to represent both cases of number:

NP → DET_SINGULAR,NOUN_SINGULAR
NP → DET_PLURAL,NOUN_PLURAL
DET_SINGULAR → a
DET_PLURAL → some
NOUN_SINGULAR → book
NOUN_PLURAL → books

If we want to further distinguish *a information* and *a book*, the grammar should be further modified as:

NP → DET_COUNTABLE_SINGULAR, NOUN_COUNTABLE_SINGULAR
NP → DET_COUNTABLE_PLURAL, NOUN_COUNTABLE_PLURAL
...

As it can be seen, the size of the CFG grows exponentially and for describing even a small subset of English grammar, the number of rules and symbols would be too

huge to be practical.

To handle such phenomena conveniently, the grammatical formalism is extended to allow constituents to have features. For the example above, we can define a feature NUMBER that may take a value of s(for singular) or p(for plural). The grammar can be rewritten as:

NP → DET(Number1), NOUN(Number2) : Number1 = Number2

instead of

NP → DET_SINGULAR, NOUN_SINGULAR
NP → DET_PLURAL, NOUN_PLURAL

Grammar with features is equivalently as powerful as Context Free Grammar, which means it recognizes the same language as CFG does. But grammar with features can dramatically reduce size to describe syntactic phenomena of natural languages such as English.

To be more understandable, features can be assigned a name, followed by a value of the feature:

NP → DET(number:V1), NOUN(number:V2) : V1 = V2

Here, the feature "number" has values V1, V2 which are variables. The grammar is conveniently defined and easy to understand.

Implementing a grammar with features can be thought of setting up a set of rules of defining features and how constituents of different features should be connected to each other. In this nature, logic programming is a good tool to implement grammars

with features. In Prolog, the grammar with features can be easily defined as:

np :- det(number:V1), noun(number:V2), V1 = V2.

In the last section, we introduced a small top down parser that uses top down search as its parsing strategy. The top down parser starts at the most abstract level (the level of sentence) and work down to the most concrete level (the level of words). So given an input string, it starts out by assuming it is a sentence, and then try to prove that it really is one by applying rules left-to-right. That works as follows: If we want to prove that the input string is of category s and we have the rule S → NP VP, then we will try next to prove that the input string consists of a NP followed by a VP. If we furthermore have the rule NP..Det N, we try to prove that the input string consists of a Det followed by an N and a VP. We use rules in a left-to-right manner to expand nonterminals we want to recognize until we have reached terminals corresponding to the words of the input string.

As we know, Prologs theorem prover itself is a depth first top down search engine, so Prolog was born a top down parser, which means in Prolog we dont need to write the parser at all, and what we need to do is to translate grammar rules into Prolog rules as we did in the last section. Simplicity is the best feature of top down parser. In the other hand, we may need to pay a price for the simplicity. Lets take a look at the following grammar:

S → S and S
S → NP VP
NP → you
VP → go

If we try to parse the sentence “you go” based on the grammar in a top down

manner, the parser is trapped in an infinite loop. Staring from the start symbol S, the parser will apply the rule S *to* S and S from left to right to prove the sentence consists of S and S, because this is the first rule it meets for S. After rewriting S to S and S, the first nonterminal is S again. The parser will do the same thing as the last step and expand the string to S and S and S. Again, it has to expand S. Thus, the parser is trapped into an infinite loop that the start symbol S is infinitely expanded without reaching any string consisting of only nonterminals. To solve the infinite loop problem in this case, we can rearrange the ordering of the rules for S as

S → NP VP
S → S and S
NP → you
VP → go

Now when S is expanded, the rule S → NP VP is applied first and in this case NP is rewritten as you, VP as go. The sentence "you go" is correctly recognized by the parser.

From linguists point of view the two grammars above are equal, which means they have the same declarative meaning. But from programmers point of view, they have different procedural meaning due to the nature of top down search. So the dilemma between the declarative meaning and procedural meaning is one of the major disadvantages of Top down parsing. A good parser should not let grammars with the same declarative have different procedural meaning.

A general solution to this problem is the Bottom Up Parsing. The basic idea of bottom up parsing and recognition is to begin with the concrete data provided by the input string — that is, the words we have to parse/recognize — and try to build bigger and bigger pieces of structure using this information. Eventually we hope to put all these pieces of structure together in a way that shows that we have found a sentence. So the bottom up parser uses rules in a right-to-left fashion. In the example above, the parser starts with the words "you" and "go", and finds out an NP and VP

from these two words by applying rules NP..you and VP..go from right to left. Again, it finds out NP and VP can form an S, which is the symbol we are looking for. The bottom up search strategy starts with something it already has instead of guessing as the top down parser. So no matter how rules are ordered, and written, the declarative meaning and procedural meaning of the grammar are always the same.

2.4 Chart Parser

The main different between top-down and bottom-up parser is the way the grammar rules are used. But unfortunately, such simple implementation would be prohibitively expensive as the parser would tend to try the same matches again and again, thus duplicating much of its work unnecessarily. That is to say, whenever there is a failure in applying a rule either from left-to-right or from right-to-left, the parser will start over again from the point of failure throwing away all the information it has got in the middle. The worst case would be the parser has used all the rules, searched all possible matches and still cannot recognize the sentence. In this case, the computational complexity is exponential.

To avoid this problem, a data structure called chart is introduced that allows the parser to store the partial results of the matching it has done so far so that the work need not be reduplicated. The chart maintains the record of all the constituents derived from the sentence so far in the parse. It also maintains the record of rules that have matched partially but are not complete. These are called the active arcs. The basic operation of a chart-based parser involves combining an active arc with a completed constituent. The result is either a new completed constituent or a new active arc that is an extension of the original active arc. New completed constituents are maintained in a list called the agenda until they themselves are added to the chart. Depending on the way rules are applied, the chart parser may be classified as top-down chart parser and bottom-up chart parser. A brief description of a bottom-up chart parsing is shown below.

Assign a label of position p_n and p_{n+1} at the beginning and the end of each word;

Push all words in the agenda;

Do until agenda is empty

{

Select a constituent C from position p_1 to p_2 out of the top of agenda;

For each rule in the grammar of form $N \rightarrow CN_1...N_n$, add an active arc of form $N \rightarrow \wedge CN_1...N_n$ from p_1 to p_2;

Insert C to chart;

If there are any active arc of the form $N \rightarrow N_1...N_n \wedge C$ from p_0 to p_1, add a new active arc $N \rightarrow N_1... \wedge C...N_n$ from p_0 to p_2;

If there are any active arc of the form $N \rightarrow N_1...N_n \wedge C$, then add a new constituent N from p_0 to p_2 to the agenda;

}

look in the chart for the constituent you want that spans the whole sentence;

The chart parser can be considerably more efficient than parsers that rely only on search because the same constituent is never constructed more than once. The worst case is to build every possible constituent between every possible pair of position. It has a worst-case complexity of $K \times n^3$. So chart parser has reduced from the exponential complexity of pure search based algorithm to a polynomial complexity.

2.5 WordNet

WordNet®[1] is an online lexical reference system, in which lexical information is organized based on meaning. It is inspired by psycholinguistic theories of human lexical memory.

In conventional dictionary, lexical information is put together in an alphabetical manner through a list. A word and its neighbors are spelled alike and words with similar or related meaning are scattered haphazardly.

[1]WordNet is a registered trademark of Princeton University

WordNet is different from the conventional dictionary in two respects: The organization and the structure of the lexical information.

The most ambitious feature of WordNet is its attempt to organize lexical information in terms of word meanings rather than the word forms. Words with the same or similar meaning are grouped into a synset, which is regarded as sufficient to explain the meaning of the group of words. Each synset represents a meaning and that is the most basic constructing unit of WordNet. We can say WordNet is a collection of synsets instead of words.

WordNet is not only a plain list of synsets, but there are various kinds of pointers between synsets, indicating their semantic relations. Done of the most important semantic relation between synsets is antonym, which means the two synsets with an antonym relation have opposite meanings. Another example is the meronymy, which indicates "a kind of" relation. It is a unilateral relation only between noun synsets.

The semantic relations are implemented as pointers in synsets and from this point of view the WordNet is a web of synsets (meanings) connected to on another by some types of semantic relations. It captures how lexicon is organized in humans lexical memory. Its organization and rich information of semantic relations are a particular advantage of WordNet in Natural Language Processing.

WordNet contains only nouns, verbs, adjectives and adverbs, while the relatively small set of English function words is omitted on the assumption that they are probably stored separately as part of the syntactic component of language. Synsets contain synonyms of the same category, while semantic relations can connect synsets of the same category as well as different categories.

2.5.1 Semantic relations between noun synsets

Inheritance relation is one of the most important semantic relations between nouns. Two words have inheritance relation if one word inherits all features of the other. The word "dog" inherits the word "mammal" and "mammal" inherits "animal", because the former word has all the features that the latter one has. To be more understandable, we can call it "is a kind of" relation. A dog is a kind of mammal and

a mammal is a kind of organism. We call mammal is the hypernym of dog and on the contrary, dog is the hyponym of mammal. One sense of a noun can only have one hypernym but may have multiple hyponyms. Through the inheritance relation, the noun synsets in WordNet are organized in a hierarchical fashion, like trees starting from some unique beginner synsets as follows:

{act, action, activity} {nature object}
{animal, fauna} {natural phenomenon}
{artifact} {person, human being}
{attribute property} {plant, flora}
{body, corpus} {possession}
{cognition, knowledge} {process}
{communication} {quantity, amount}
{event, happening} {relation}
{feeling, emotion} {shape}
{food} {state, condition}
{group, collection} {substance}
{location, place} {time}
{motive}

Part-Whole relation can be denoted as "a part of" relation. "leg" is a part of "body", so "leg" and "body" has a component relation, because leg is a part of body. We call "leg" as the meronym of "body" and on the contrary, "body" as the holonym of "leg". We often regard meronym as a feature of the holonym, say, "leg" is a feature of a "body". And as we introduced before that inheritance relation can let a hyponym inherit features of a hypernym, so if word A is a meronym of word B, it is also a meronym of the hyponym of word B.

Like inheritance relation, part-while relation is also transitive and can relate terms hierarchically. That is, parts can have parts: a finger is a part of a hand, a hand is a part of an arm, and arm is a part of a body. So this may produce complicated

relation that meronym of the meronym of a word is also a meronym of the hyponym of the word.

2.5.2 Semantic relations between verb synsets

In logic, entailment is properly defined for propositions; a proposition A entails a proposition B if and only if there is no conceivable state of affairs that could make A true and B false. Entailment relation between verbs resembles meronymy between nouns. For example, snore lexically entails sleep. It is a unilateral relation. So on the contrary, sleep does not entail snore. Troponym is a particular kind of entailment, in that every troponym A of a more general verb B also entails B.

The causative relation picks out two verb concepts, one causative (like show), the other resultative (like see). This relation is transitive. That is, if verb A causes verb B and verb B causes verb C, we can conclude that verb A also cause C.

The hyponym relation among verbs is almost the same as that among nouns.

There are some semantic relations that are shared by all categories. Synonymy groups a set of words with the same or similar meanings into a synset. And antonymy means the two synsets have opposite meaning.

In addition to semantic relations within categories, some semantic relations connect synsets from different categories. A descriptive adjective modifies an attribute and there is a link between the adjective and the synset representing the attribute. An adverb may be linked the adjective it is derived from.

Prolog version of WordNet is freely available, in which various semantic relations are implemented as Prolog predicates.

2.6 Frame Semantics and FrameNet

Frame semantics is a research program in empirical semantics which emphasizes the continuities between language and experience. It is based on the theory that to understand a concept it is necessary to understand the entire system of concepts. In linguistics, that translates into that a word only has meaning in a certain situation. The situation is represented by a frame and a frame consists of a set of frame elements

which can be used to assign thematic roles. The notion of frame can be exemplified with the commercial transaction frame, whose frame elements include seller, buyer, goods, money and etc. Semantically related words can evoke the same frame. In the last example, verbs like sell, buy, pay, cost or charge may evoke the commercial transaction frame. Knowing the meaning of these words requires knowing the whole situation of a commercial transaction.

Although similar notions have developed and are employed in other fields, particularly artificial intelligence and cognitive psychology, such as Minskys frame in AI, schank and Abelsons script, the notion of frame used in frame semantics can be traced most directly to Fillmores case frame in linguistics. Case frame were mentioned as "characterizing a small abstract scene or situation so that to understand the semantic structure of the verb, it is necessary to understand the properties of such schematized scenes". The verb is placed as the central role of frame semantics. And a complete description of these verbs must also include information about their grammatical properties and the various syntactic patterns in which they occur.

Take "Lei bought a book from Zhang" as an example, the same commercial transaction frame can be also realized as "Zhang sold a book to Lei".

The FrameNet is a database of frame semantics description of several thousand English lexical items and backing up these descriptions with semantically annotated attestations from contemporary English corpora. These descriptions are based on hand-tagged semantic annotations of example sentences extracted from large text corpora and systematic analysis of the semantic patterns they exemplify by lexicographers and linguists. The database has three major components: lexicon, FrameNet database and annotated example sentences. Lexicon is composed of conventional dictionary data for human readers, formulas about how frame elements are realized syntactically, links to semantically annotated example sentences and links to the FrameNet database and other machine readable lexical resources such as WordNet and COMLEX. FrameNet database contains descriptions of each frames based conceptual structure and gives name and descriptions to frame elements. Annotated example sentences are the major part of the whole FrameNet. They are semantically tagged to exemplify the semantic and morphosyntactic properties of the lexical items.

CHAPTER 3

SEMANTIC PARSING

The semantic parser is a tool used to analyze the semantic structure of a natural language sentence. As we all know, Chomskys theory formally defines a language as a set of strings over an alphabet. He defined the grammar as a set of rules to compose the alphabet into valid strings of a language, and the classified languages into categories. Usually the natural language can be classified as context free language since the context free grammar can describe most of the structure in natural language. A syntactic parser is a computer program to parse a valid natural language sentence into a parse tree indicating how the sentence regarded as a string can be decomposed into smaller syntactic constituents. Correspondingly, the purpose of a semantic parser is to analyze the structure of sentence meaning. The meaning of a sentence is decomposed into smaller pieces of semantic units connected with various semantic relations, and the parser will represent the structure, including semantic units as well as semantic relation connecting them in a formal format.

3.1 Semantics

Semantics has been well studied in many disciplines and it is defined from many prospective, such as psychology, cognitive science, linguistics and etc. But definitions in these areas are highly human oriented, and usually based on several assumptions about human beings. In order to let computers handle semantics, a more general definition is needed that makes no distinction between human and machines.

Semantics is also called meaning or sense. Semantics lies on a medium and should be understood by an agent. The medium carrying semantics can be anything that carries information. A gesture, a sign, a flash of light, a piece of code can all potentially carry semantics, but it has to be understood by an agent. An agent is an entity that can receive input and produce output. The agent receives the medium as input

and produces output as its reaction and this process between input and output is understanding. Without understanding by the agent, the medium carries no semantics and it is meaningless. Before the process of understanding, the medium has to be transformed into an internal format or representation that the agent can handle and correspondingly the output of the agent can be used to trigger various reactions. The understanding process may rely on rules, knowledge or experience to properly handle its input and produce output. The whole procedure is illustrated by the figure:

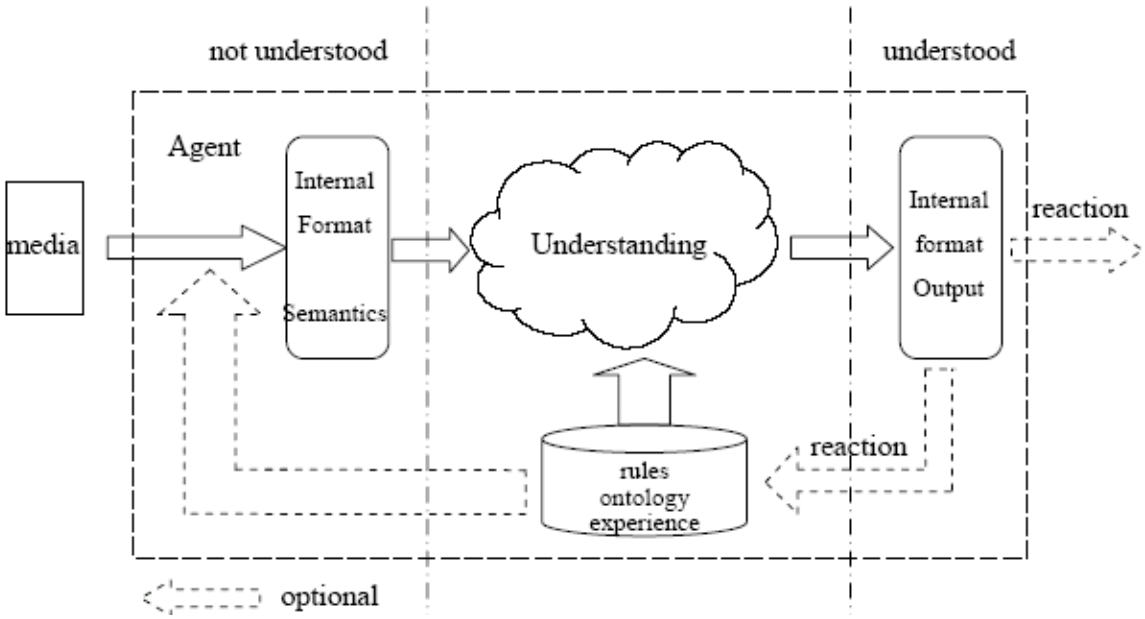

Figure 3.1: Definition of semantics

For the agent, the semantics of the medium is its internal representation which can be accepted as input by the understanding process and the transformation from the medium to its internal representation is semantic interpretation. The reaction does not necessarily mean physical reaction. It may also include change of knowledge, addition of experience or any other possible operations that the agent is capable of.

We can define two levels of semantics and understanding. As long as the media can be transformed into an internal representation in the agent, we call it level one semantics or shallow level semantics. Accordingly, the understanding is level one understanding and shallow level understanding. If the media can be transformed into

an internal representation and can further invoke a reaction based on the processing rules, we call it level two semantics or high level semantics. Accordingly, its understanding is level two understanding and high level understanding. The high level is more restrictive than the shallow level.

We can take the DVD player as an example to illustrate what is medium, semantics, and understanding. When a person hit the play bottom on the players panel, the physical contact between the finger and the button is translated into an electrical signal. Corresponding to the figure, the physical contact is the medium which is turned into DVD players internal representation as an electrical signal. The DVD player "knows" how to handle this signal since its designer has incorporated processing rules in the electronic boards. These rules translate it into another set of signals which invokes a set of reactions, reading video data from the disc and displaying its content on the screen. By now we can say the DVD player understands your touch of the button by providing a reasonable reaction and the semantics of that for the agent is the electrical signal the button triggers. If you dont touch any button, you any other actions do not have semantics to the DVD player because they can not be transformed into electrical signals that the processing rules can process.

3.2 Semantics of languages

Human, as a very intelligent agent, has much more processing rules and knowledge than a DVD player. Human beings share a large collection of knowledge and experience that represent the world. The natural language is often used as a medium to represent humans knowledge, ideas and carries semantics for communication.

A language is defined as a set of strings over an alphabet. Strings of the natural language are used to represent facts, concepts, events, etc. in the world. In order to understand what the strings represent or extract semantics of the strings, humans need to transform the strings into its corresponding internal representation, called semantic representation, so that this representation can be used to draw conclusions, triggers physical reactions, update knowledge and etc.

According to the definition of semantics from the previous section, the semantics of

language can be classified as level one semantics and level two semantics. For example the string "apple and banana" means two entities. Though it may not invoke any reaction for a human, it is still interpreted into an internal representational format and hence carries level one semantics. For the string "John will marry Linda", if the agent is Lindas friend and he hears about it the first time, this is first transformed into its internal representation and based on his knowledge it may trigger his reaction (being happy, or sad, etc). In this case, the string carries high level semantics to Lindas friend. "Colorless green dreams sleep furiously" is a very good example to illustrate the distinction between the two levels of semantics. We know what it talks about but it does not make any sense to us because according to our knowledge and common sense we can not understand it and provide a reaction to it. So this only carries shallow level semantics. This thesis only takes care of the level one semantics because for level one semantics we only need to know how to transform strings in a language into its semantic representation, while for level two semantics we need every large collection of knowledge or common sense to understand it and how the knowledge is represented in human mind is still in debate.

A major usage of language is communication. An agent uses language to represent his beliefs, ideas, and the semantics of the language can be extracted by other agents. Successful communication between agents must meet the following two criteria:

1. They must use the same language. That is to say, they should use the same set of strings to represent semantics since languages are the media of semantics. An English speaker can not understand the meaning expressed in Chinese.

2. The same string carries the same semantics for all agents in communication. The communication will not be successful if one side says "apple" but the other side interprets it as a "banana".

The natural language is an exclusive tool for human kind to represent the knowledge of the world and communication due to humans superior intellectual capacity. Based on the above two criteria, in order for computers to communicate with agents such as humans, or to understand human languages or a subset of them, they need human knowledge.

3.3 Word meaning and sentence meaning

Natural language is a set of strings over an alphabet. How do strings represent facts, concepts and events for an agent? If we define K as all semantic representations accepted by humans understanding system, and for every semantics k_i , there is a unique corresponding string s_i , which can be mapped to k_i by the function F such that $F(s_i) = k_i$. The set S of all s_i is a language that can be used as the medium to carry the semantics to represent the set K for the agent and we define it as $F(S) = K$.

If the semantic representation of the fact "the meal is good", for instance, is k_i, we can use an arbitrary string such as "almlr" to represent the k_i. For another fact "the meal is bad", as k_m, we can also use such arbitrary string "bscad" to represent k_j. Then the function F is

$$F = \begin{cases} k_i & \text{if } s = \text{“}almlr\text{”}; \\ k_j & \text{if } s = \text{“}bscad\text{”}. \end{cases}$$

Using strings to represent semantics in this way can be called one-to-one mapping. That is, a unique string represents a unique semantics and there is no relation between the semantics and the composition of the string. In this way, if there are n pieces of semantics we need n strings and a function F with has n entries of mappings. Using a language in this way has an obvious disadvantage for human that human has to memorize all these strings individually for each semantic representation of facts, concept or events in the world, even though many of them is rarely used or highly inter-related. In the example above, the two facts "the meal is good" and "the meal is bad" are related in that they both talk about quality of the meal, and the only difference is the value of this quality. But the composition of their corresponding strings "almlr" and "bscad" does not reveal the semantic relatedness. Secondly, if the fact "the meal is good" is not referred to very frequently, memorizing a dedicated string for it is not memory efficient.

A better way, which is adopted by the natural language, is to use one-to-one mappings to a relatively small number of facts, concepts and events which are referred to frequently, and a set of semantic rules to compose some of these small semantic

units into a big whole. The advantage of it is that a human only needs to know a small finite number of strings and a finite set of rules and is still able to use them to compose infinite number of strings representing infinite number of facts, concepts and events in the world ('the generative lexicon' by James. Pustejovsky). This makes the distinction between words and sentences. Morphologically words and sentences are all strings, but a word, as the smallest unit of string denoted as w_i, is used to represent those frequently referred semantics k_i; while a sentence t_i, is composed with words by syntactic functions G. The set of all words is vocabulary V and the set of all sentences T is the natural language L.

Since all sentences in the natural language can be generated by some words in the vocabulary and some grammatical functions, we can write $T = G(V)$ and $Number(T)$ is the number of sentences. A natural language can be formally defined as follows:

$$L = G(V) \begin{cases} V = \{w_1, w_2, ..., w_n\}; \\ G = \{g_1, g_2, ..., g_m\}; \\ Number(G(V)) = \infty. \end{cases}$$

This definition tells us that a natural language consists of a vocabulary V with a finite number of words, and a finite set of grammatical rules to compose an infinite number of sentences to be able to represent infinite number of facts and events in the world.

Correspondingly, the semantics of the natural language can be defined as follows:

$$W = R(K) \begin{cases} K = \{k_1, k_2, ..., k_n\}; \\ R = \{r_1, r_2, ..., r_m\}; \\ Number(R(K)) = \infty. \end{cases}$$

The focus of the thesis is to describe the function R and represent the semantic structure of the natural language sentence in a computer understandable format.

3.4 Literal meaning and contextual meaning

In the previous section about the semantics of languages, the semantics is classified into shallow level semantics and high level semantics. The high level semantics should be understood according to the agent's ontology. Hence in many cases the high level semantics is understood in some context. A typical example is the reference resolution in "I told you that story", in which "I", "you" and "that" refer to something in the context. A sentence's literal meaning may not be the same as its contextual meaning. For the question "do you know what time it is?", its literal meaning is to ask if the addressee knows the time. The answer to this question normally should be "yes, I know" or "no, I don't". Now suppose the question is asked in an airport by a traveler to an airline clerk. The clerk won't interpret it by its literal meaning. Instead, in this context the semantics of this question is similar to "what time is it?".

In some cases, the literal meaning and contextual meaning can be different dramatically and the contextual meaning may vary in different context. Consider the following two scenarios:

Scenario I: John and Mary had a blind date and before they left each other, Mary said, "I like you, John".

Scenario II: John has been chasing Mary for three months and he asked Mary one day, "Mary, do you love me?" Mary said, "well, I like you, John."

In the second scenario, the "I like you" may imply an opposite contextual semantics to its literal semantics than that in the first scenario. Contextual meaning also includes metaphor.

The semantic parser will not parse the sentence meaning based on its contextual semantics. One reason is that understanding contextual semantics requires very high level ontology and common sense while research in high level knowledge representation is still in its infancy. Another reason is that this provides flexibility for users so that they can use their own way to incorporate contextual information. They can use their

own ways to resolute reference, to find out contextual implication, etc.

3.5 Syntactic relations and semantic relations

In previous sections, we have defined a natural language sentence as a string consisting of a set of syntactically related words. Then how does human brain represent the meaning of the sentence? Does it remember the meaning or the wording of the sentence?

A clever experiment by Wanner(1974) examined meaning retention versus surface form. Wanner avoided bringing to psychological experiments special strategies that are not representative of language processing under more natural circumstances by giving the participants fairly routine instructions to an experiment, then giving them a surprise test on the instructions themselves. The instruction was:

"When you score the results, do nothing to your correct answer but mark carefully those answers which are wrong."

After hearing the sentence, the participants were tested on one of it. Some were tested on the wording "your correct" and were given a recognition test with the choice of the original wording and "correct your", which change the meaning of the sentence. Others were tested on their ability to distinguish between "mark carefully" and "carefully mark", which mean the same thing. Wanner found excellent memory for meaning, 100% correct on "your correct", but only half performance on wording, 50% correct on "mark carefully".

This experiment shows that people tend to retain meaning (semantics) instead of the exact wording(syntax), and the semantic structure is not always the same as the syntactic structure. Syntax can be used as a tool for understanding and realization of semantics. That is to say, syntactic rules do not have a one to one correspondence to semantic rules. One reason for that is there are human tends to use a simplified grammar for a language. A very complex grammar may make the language very

hard to learn, hard to use, prone to errors and hence may discourage the use of the language. Another reason is that human has the capability to easily figure out the semantic relation from the syntax with the help from other resources including ontology, lexicon and etc.

As seen in the definition of semantics in figure 3.1, the semantic representation should be independent to the medium, which means it is possible to represent the semantics independent of grammar.

3.6 Compositionality of semantics

The general principle of the compositionality of semantics is that the meaning of the whole is the function of the meaning of the parts. Meaningful components correspond to syntactic constituents.

syntactic rule: $S \rightarrow C_1, C_2, ..., C_n$,S and C_i are syntactic constituents
semantic rule: $S' \rightarrow F(C'_1, C'_2, ..., C'_n)$,S' and C'_i are corresponding semantics.

Compositionality is a very important principle in analyzing semantics of a sentence. We can use a top-down approach from the whole semantics of the sentence and recursively decompose it into smaller semantic constituents until reaching the smallest semantic unit, lexicon. Analyzing semantics in this way is very much like syntactic analysis and the output is similar to a tree. The difference is the analysis includes semantic roles, which are not merely concatenation, as the syntactic roles are.

3.7 Information used in determining semantic relations

3.7.1 Syntax

The syntax provides us with most of the clues about the semantic structure of a sentence. According to the principle of semantic compositionality, a syntactic constituent

carries a relatively complete and independent unit of semantics, so the syntactic parsing into smaller syntactic constituents is necessary. In the other hand the syntactic relation can imply, and in many cases directly identify, semantic relations. Although for efficiency consideration, the number of syntactic rules has to be small in order to make the language easy to use, it is still the major tools for human to figure out or establish semantics structures.

3.7.2 Lexicon

The lexicon is divided into content words and function words based on their syntactic functions. Content words carry semantics representing facts, concepts and events. Content words include nouns, verbs, adjectives and adverbs. Content words are open, and new words are being created constantly. The semantics of individual content words will not be handled by the semantic parser, because the parser is used to analyze the structure of the semantics, but not the semantics itself. It should be noticed that morphological change of a content word may constitute additional semantics and the relation between its original semantics and the additional semantics should be studied. For example "books" is the plural form of the word "book". The morphological change from "book" to "books" provides the quantity. This additional semantics and the original semantics of "book" form an attribute modification relation.

All words other than content words belong to function words, such as preposition, determiners, auxiliary verbs and etc. Their major function is to help identify syntactic and semantic relations. Some of them carry semantics while some of them dont. Function words play important roles in composing semantics in content words and they should be paid close attention.

The lexicon is classified into several categories based on words syntactic features. But the number of categories is relatively small due to the simplicity of syntax that they do not reveal much semantic information. Based on semantic relations between synsets in WordNet, we can assign semantic features to the syntactic category. In addition to the formal syntactic categories of English, we created some new categories such as "number", "unit" based on words semantic functions.

Phrase words like “a little bit”, “make up for” should be treated and compiled in the lexicon as single words because they represent single semantics though morphologically composed of several words.

3.7.3 Ontology

Since the syntactic relations and semantic relations do not have a one-to-one mapping, and there is usually a smaller number of syntactic relations, the same syntactic relation may represent different semantic relations in different situations. So the situation needs to be considered in order to figure out the semantic realization of the syntactic relations. “Situation” is an ontology level concept and we need a shallow level ontology database that can be use to identify, distinguish situations and define how semantic relations are realized by syntactic relations in these situations. The solution to this used in the semantic parser is FrameNet. FrameNet is regarded as an ontology database based on the theory of frame semantics. The frame semantics thinks that a sentence tells a situation, denoted as “frame”, and various syntactic constituents in the sentence play semantic roles in the situation, denoted as “frame elements”. These roles indicate the semantic relations in these syntactic constituents in the sentence. For example “Alice paid 30 dollars for the toy bear” tells a commercial_transaction situation (frame) according to the FrameNet. The semantic role “Agent” is assigned to “Alice”, the subject of the sentence, indicating who initiates the transaction. “Paid” is the action verb that triggers the situation (frame). The role “Money” is assigned to “30 dollars”, object of the verb “paid”, as the expense for the “Agent” in the transaction. The “toy bear” is the “goods” role in this situation (frame). FrameNet has defined most frequently used frames to represent various situations and most importantly it defines in each frame, how frame elements are syntactically realized. This explains why the syntactic relation can represent different semantic relation and gives us a way to find it out. But it should be noticed that FrameNet is a shallow level ontology. It does not contain such high level knowledge that a ball is round or the sun rises in the morning. It is very useful for us to analyze the semantic structure of a sentence, but it is very linguistics oriented and its ontology

should normally be applied in one sentence and hence it can not be used to analyze contextual meaning which need very high level ontology and common sense.

In addition to knowledge about situations and what semantic relations the syntactic relations play in these situations, knowledge about word relations is also necessary. According to the principle of compositionality, the semantics of the whole sentence can be decomposed to the smallest semantic units, words. To figure out how word semantics can be composed into bigger semantic constituents, we need knowledge about their semantic relations. The semantic relations among words can be within the same category or across categories. WordNet is the lexical database providing rich semantic information. The WordNet indexes content words based on their senses rather than in an alphabetical order. According to the word sense, the WordNet connect semantically related words through links indicating some semantic relation. The link can be within the same category or across categories and it represent various semantic relations such as "synonym","antonym","part-whole" relation, "isa" relation and etc. Such information about semantic relations between words enable us to find out how word semantics can forms bigger semantic constituents. If we say FrameNet provides sentence level ontology, then WordNet provides lexical level ontology.

3.8 Semantic structures and the method of parsing

In this section, various semantic relations between semantic constituents are defined. According to the principle of semantic compositionality, the definition starts at the sentence level and in general follows a top-down approach to lexical level. Since the semantic analysis is based on three types of resources: Syntax, Lexicon and Ontology, with these relations we describe what the method to find out the semantic relations.

The general architecture of the sentence semantic structure is called role+modifier architecture. We think of the semantics of a sentence as a situation or scenario, in which there are various semantically related participants. Each participant plays a role in the situation or scenario and they are further described by their modifiers. The modifiers for each participant are both syntactically and semantically local to that participant and they wont interfere with those for other participants.

The verb is regarded as the key element that triggers the frame (situation), and frame elements (roles) are assigned to syntactic constituents according to the syntactic realization rules in the frame. In other words, the verb tells us the semantic relations among syntactic constituents by triggering a frame. The entities acting as roles can be further modified by various modifiers. If the syntactic constituents assigned a role contains a verb phrase, which means it is an independent situation itself, it will trigger a new frame for the role and its child syntactic constituents are assigned roles according to the new frame. This procedure will apply recursively until a syntactic constituent can not trigger a frame.

The role+modifier architecture can be defined as follows:

$$S' \rightarrow [verb, [modifier_1], [modifier_2], ...], [role_1], [role_2], ...$$
$$Role \rightarrow [verb, [modifier_1], [modifier_2], ...], [role_1], [role_2], ...$$
$$Role \rightarrow head, [modifier_1], [modifier_2], ...$$

The modification relations are defined as follows:

The formation of the following semantic relations is highly related to the structure of human lexical memory. Similar to the mechanism for the use of words and sentences, human mind only lexicalize those frequently used facts, concepts and events. Lexicon in the human mind is highly interlinked through various semantic relations that form a semantic network. The similarity in the mechanism of lexicalization in human minds and natural language sentences indicates the similarity in semantic relations of lexical units in human minds and sentences.

If we think the world is composed of entities and interactions between entities, the attributes of the entities and their interactions are the lexical units describing what a perceiver observes about them. Theoretically, if all entities and interactions as well as all their attributes and values of the attributes are individually lexicalized, the attributive modification relation that connects the value of an attribute of a specific entity or interaction is powerful enough to represent everything a perceiver observes. Usually entities in the world are lexicalized as nouns and their interactions

as verbs. The values of their attributes are descriptive adjectives and descriptive adverbs respectively. Examples are “delicious meal”, “move slowly”. The attributes of entities or their interactions are usually gradable. But not all grades of the attribute are lexicalized, so there exists degree modification that can modify the grade or degree of a descriptive adjective or descriptive adverb. An example of this is “very good”.

Attributive modification relation could be universal if every entity and interaction is lexicalized individually, which means every single thing* is lexicalized differently. For example, there are two apples, but you cant call both of them “apple”. Instead, you should use totally different words to represent them. Obviously this is a very inefficient way. A solution to this is to lexicalize things with similar or the same attributes and frequently used as one lexical unit, and use referential modifiers to designate individual instances. Usually some function words such as “the”,“that” or occasionally referential adjectives like “former”,“alleged” can be used as referential modifiers. Syntactic constituents such as preposition phrase attachments and some adjective clauses can also play this function. Using referential modification can save a lot of lexical memory in that instances of the same type of things will not be individually lexicalized.

Referential modification allows things with similar attributes not to be lexicalized repeatedly. But for verbs and nouns, which represent entities and their interactions in the world, their structure in humans lexical memory is not one dimensional. Instead, its rather a hierarchical structure. Based on attributes of the things they represent, lexical units are further generalized into a more abstract unit, a parent lexical unit, which can represent all its child lexical units. The child words inherit attributes of the parent word. For example, “fruit” is the parent word of “apple”, “banana” and “peach”, because “fruit” has some attributes that all its child words have. In the sentence “Fruits are edible”, the term “fruit” can represent everything its child words can represent, such as “apple”, “banana”, “peach” and etc, so that we dont need to enumerate in such way “apples, bananas, peaches, oranges,..., are edible” in order to represent the same meaning. The advantage of this mechanism is obvious. The restrictive modification is used to represent unlexicalized things that inherit attributes of a lexical unit but intended to mean additional attributes. An example is “computer

desk". It has all attributes of a desk and it is a desk. But it is not lexicalized as a word, and a noun "computer" to put a restriction on the "desk" that it is used for computer operations. In addition to nouns, restrictive adjectives also can be restrictive modifiers, such as "military technology". A distinction between restrictive adjectives and descriptive adjectives is that descriptive adjectives have degree and comparative form. "very good boy" is valid but "very military technology" is invalid because "military" is not an attribute of technology but just a restriction to a smaller domain put on the term "technology".

Everything in the world can be represented by a string in natural language. The string could be a word like "apple", or a phrase "that computer desk", or even a sentence "he won the game yesterday". But it is not required that everything should be represented by only one string. It is possible to refer to the same thing with more than one different string. In the sentence, "John is my brother", John and my brother refers to the same person. You may represent this person with "my brother" or just simply call him "John". The semantic relation between the two strings representing the same thing is co-reference relation.

Coordination relation exists between parallel syntactic constituents. The parallel constituents are connected with coordinating conjunctions such as "and", "or", "but", "bothand", "eitheror", "than", and etc. Based on the conjunction word, coordinate relation can be subcategorized into additive, selective, adversative and comparative. The constituents they connect vary from words to sentences.

Complement is usually an attachment to provide additional description on its head words, which can make the sentence more coherent and concise.

Sentence, as a semantic unit that carries a coherent, independent and relatively complete piece of information has functions. Sentence function refers to the purpose of the sentence meaning. Sentence functions can be assertion, asserting a fact or ones belief, yn-query, asking for confirmation of addressers belief, general query, asking for an object to be identified, command, describing an action to perform, exclamation, indicating addressers amazement. Sentence function is an important factor affecting addressees reaction. Therefore, it should be included in sentence semantics and can be identified by syntactic pattern of the sentence.

3.9 Data, tools used and general procedure

In order to identify the semantic structure of the sentence, ontology, syntactic features and lexicon are needed and they should be ready before the semantic parsing process begins.

FrameNet is the resource for sentence level ontology to identify frames and assign semantic roles. It comes as a corpus database, where sentences are both syntactically and semantically annotated with a frame and its syntactic constituents are assigned semantic roles. We can extract the syntactic and semantic features and formalize how semantic roles are syntactically realized in a more program readable form. How the features are extracted from the annotated corpus and formalized will be described in detail in later sections.

WordNet is the resource for lexical level ontology to identify those structures that need lexical level semantic relations. All necessary semantic relations are extracted from WordNet and written as features for words so that they can be directly used in the parsing process.

The syntactic parser provides the syntactic features for semantic analysis. Some syntactic features can be used to directly derive semantic relations and some are used to assign semantic roles. If they are used for semantic role assignment, the set of syntactic features should match those used in FrameNet.

After the data and ontology are properly prepared, the general procedure of the semantic parsing is divided into three steps. 1. syntactic parsing into an intermediate format. 2. semantic role assignment 3. Processing global rules.

3.9.1 Compiling the Lexicon

Words are uniformly defined as lex/2 predicate in Prolog. The word is assigned features including syntactic feature and semantic features indicating what syntactic and semantic functions it plays. For example, for the word “dog”, its features include its category (noun), number(singular) and countability(countable). In table 3.2, features for categories of content words are listed and described.

3.9.2 Rules learned from FrameNet

FrameNet is the ontology database based on the theory of frame semantics. It defines most frequently used frames and syntactic realization of frame elements by annotating a large collection of text. Since it is a corpus based database and its format can not be directly read by Prolog, we need to extract the syntactic features and the semantic annotation in the text and formalized into a Prolog readable format. The syntactic features selected are the Phrase Type(PT), Grammatical Function(GF), Relative Position to the target word, the Mood(active or passive) of the sentence, if the phrase type is prepositional phrase, the preposition is extracted. The sequence of the role is also kept.

The following is a FrameNet annotated sentence:

I	had CHASED	selden	over the moor
[Theme]	Target Word	[Goal]	[Path]

Its syntactic and semantic features are listed in table 3.3:

The corresponding formalized rule for this is:
[active,[ext,np,before,theme],[obj,np,after,goal],[comp,pp,after,over,path]]

In FrameNet, there are multiple annotated sentences for each frame to demonstrate multiple possible syntactic realizations. All of them are collected and put into a list for that frame.

3.9.3 The feature augmented syntactic analyzer

This is the first phase of semantic parsing. It accomplishes three tasks:

a) check if the sentence if grammatically right.

It is a full featured syntactic analyzer based on feature augmented context free grammar. Syntactic constituents are assigned features, and the grammar is a set of rule defining constituents with what features and what value can be combined to a bigger constituent.

b) identify some semantic relations.

Some semantic relations can be identified in this phase. These semantic relations, such as attributive modification, restrictive modification and etc. include those that can be identified based on lexical morphological rules, lexical relations or syntactic relations.

c) provide syntactic features for semantic role assignment in later phase.

Target words are annotated and syntactic features of other syntactic constituents are also annotated in the same format as its definition so that they can be matched to its frame definition. The target words and syntactic features are used for frame identification and semantic role assignment in the second step of semantic parsing.

The output of this phase is an intermediate output that contains some semantic relations and syntactic features. For example:

"He kicked the dog"

The output for the sentence of this phase is:

[tag, ext, np, before, [[entity, [he], reference(third)], [modification, quantity(single)], [modification, gender(male)]]], [target, v, kick, active, [kick]], modification, time(past)], [tag, obj, np, after, [[modification, [the], reference(definite)], [target, n, dog, [dog]]]]]

3.9.4 Role assignment

The major task of this phase is to assign semantic roles. First, the target word that triggers the frame is identified based on the annotation provided by the output of the last phase. Then, the FrameNet defined rules for that frame is located and the

syntactic features in the rules and those extracted in the last phase are matched to find which the correct role is for a specific syntactic constituent. This procedure is applied recursive from sentence to syntactic constituents until all target words which trigger frames are processed.

3.9.5 Applying global rules

For most function words, their semantics is static and wont change in different situation. But some function words, especially prepositions, their semantics may vary. For example:

"I bought a book on semantics."
vs.
"I saw a pen on the table."

The preposition on usually indicates on location above a surface, which is used in the second sentence. But in the first scenario, it refers to a topic. This example demonstrates that the default or usual semantics may change in some special situations. Accordingly, we define default semantics for the word as a global rule and assume that its special use is defined in the frame in FrameNet, which can be identified in the last phase. So this phase is to use global rules to identify semantic relations that were not identified in the last two phases.

3.10 Final output

According to the principle of compositionality, the semantic structure of a sentence is analyzed in a top-down manner and hence the format of the output is a recursive structure, like a tree.

If we use the semantic parser to parse the sentence:
"He kicked the old dog"

The output of the semantic parser on this sentence is

[Agent [[entity, [he], reference(third)], [mod_a, quantity(single)], [mod_a, gender(male)]]], [action [kick]], [modification, time(past)], [victim [[modification, [the], reference(definite)], [mod_a, age(old)],[entity, [dog]]]]]

The semantic parse tree is:

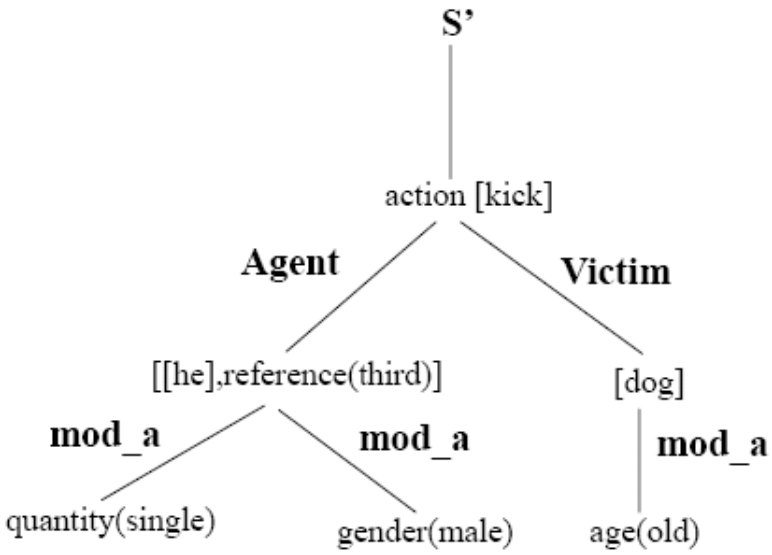

Figure 3.2: Semantic tree of "He kicked the old dog"

Name	Syntactic Pattern	Examples
Attributive Modfication	adj_d noun	a good boy
	adv_d verb	walk slowly
	noun pp	boy on the board
	verb pp	play on the board
	verb infinitive	play to win
	verb adv_clause	you can win if you work hard
Restrictive Modification	adj_r noun	military technology
	noun noun	soccer ball
	adv_r verb	parse semantically
	adv_r adj	theoretically wrong
Referential Modification	det noun	that boy
	noun adj_clause	the team that won
Coordination	...(and,or,but,than)...	boys and girls
	both,either,neither	both I and you
Complement	noun adj_clause	He, who didn't make a mistake, is not responsible for this accident
Degree Modification	adv_dgr adj_d	very good
	adv_dgr adv_d	extremely slowly
	number adj_d	two years old
	number adv_d	two times faster
Sentence Function	assertion	I like playing soccer
	yn-query	Do you like playing soccer?
	query	what do you like playing
	command	eat as much as possible
	exclamation	what a good story!
		how interesting it is!

Table 3.1: Semantic relations and their syntactic realization

Noun:

Feature	Values	Description
Number	singular/plural	to distinguish singular and plural form
Countability	countable/uncountable	to distinguish countability of nouns

Verb:

Feature	Values	Description
Transitivity	transitive/intransitive double transitive	to distinguish transitivity of verbs
Form	normal/infinitive/ present_participle/ past_participle	the form of a verb can be used to identify its semantic function

Adjective:

Feature	Values	Description
Type	descriptive/restrictive/ referential	to identify descriptive, restrictive and referential modification
Attribute Degree	arbitrary base/comparative/ superlative	the attribute the adjective modifies the degree of the adjective

Adverb:

Feature	Values	Description
Type	descriptive/restrictive/ referential/degree	to identify descriptive, restrictive, referential and degree modification
Attribute Degree	arbitrary base/comparative/ superlative	the attribute the adverb modifies the degree of the adverb

Table 3.2: Features for content words

	I	had chased	Selden	over the moor
GF	Ext		obj	comp
PT	NP	Taget word	NP	PP
Position	before		after	after
Voice		active		
Preposition				over
Role	Theme	Goal		Path

Table 3.3: Example of features extracted from FrameNet

CHAPTER 4

APPLICATIONS

The semantic parser can provide the semantic structure of a sentence presenting how the whole sentence meaning is composed with small semantic units with rich information on their relations. The advantage of it over syntactic parsing is that we can access semantic units of a sentence in a more meaningful way. Tracing back to figure 3.1, we know that in order to understand semantics, knowledge and rules in the understanding system is necessary to provide meaningful reactions. In most cases, the reaction is based on correct knowledge about the world. Semantic relations are closer to common sense about the world than syntactic relations. And hence semantic parsing moves the understanding of natural language to an upper level.

The semantic parser can be used in various fields of Natural Language Processing. In the following several sections, possible applications of the semantic parser in these fields are described.

4.1 Information Extraction

Information Extraction is the process that selectively salient facts about prespecified types of events, entities or relationships, explicitly stated or implied, in one or more text. These facts are then usually entered automatically into a template or database, which may then be used to analyze the data for trends, to give a natural language summary, or simply to serve for on-line access. A template defined by either human or computers consists of labeled slots and rules were provided on how the slots are to be filled.

Below is an example information extraction task. The task is to fill the template with information about succession events extracted from the text.

New York Times Co. named Russell T. Lewis, 45, president and general manager of its flagship New York Times newspaper, responsible for all business-side activities.

He was executive vice president and deputy general manager. He succeeds Lance R. Primis, who in September was named president and chief operating officer of the parent.

The template filled by the information extraction process could be:

<ORGANIZATION-1>
NAME : "New York Times Co."
<ORGANIZATION-2>
NAME : "New York Times"
<PERSON-1>
NAME : "Russell T. Lewis"
<PERSON-2>
NAME : "Lance R. Primis"
<SUCCESSION-1>
ORGANIZATION : <ORGANIZATION-2>
POST : "president"
WHO_IS_IN : <PERSON-1>
WHO_IS_OUT : <PERSON-2>

Information Extraction could be of great significance to information end-user industries of all kinds, especially finance companies, banks, publishers, and governments, which may need to find facts like computer take-overs, terrorism events or etc. from widely scattered text.

Information Extraction technology has not reached the market yet. One of the biggest difficulties is that there are many ways of expressing the same fact.

David sold his company to Stanly.

Stanly bought Davids company.
Davids company cost Stanly 3 million dollars.

The three utterances have different syntactic structures while expressing similar meaning. From this example, to effective extract facts that end-users want, it is necessary to understand the text.

4.2 Machine Translation

Machine Translation is the attempt to automate all or part of the process of translating from one human language to another. It is one of the oldest applications of Natural Language Processing. From the earliest days, MT has been bedeviled by grandiose claims and exaggerated expectations. Some groups took a "brute force" approach and based their work on the assumption that it was desirable, and indeed necessary, to come up with a working system as soon as possible, to take advantage of the enthusiasm that reigned at that time. They saw word-for-word translation systems as an adequate starting point that could be improved using feedback from the output. The basic need was for larger computer storage, and it was not felt necessary to solve all the foreseeable problems before trying to develop a system.

Despite all this enthusiasm, however, major problems began to appear. Translators and linguists considered it impossible to define language in a manner rigid enough to produce good translation. Early, over-enthusiastic claims and promises about the prospects of "automatic translation" eventually led the U.S. government, in the early 1960s, to commission a report on the state of play: the now infamous ALPAC report (ALPAC, 1966) was hugely critical of MT as a whole, concluding that MT was more expensive, slower, and less accurate than human translation, and that there was no immediate or eventual prospect of usable MT. It was widely accepted at that time that automating the translation process was forever doomed to failure. Part of the problem was that these early MT systems were based on the idea that translation

could be achieved basically by word-for-word substitution with some local adjustments. But characterized as the use of computers to perform translation of natural language texts, with or without human assistance, MT has to deal with almost every imaginable problem in computational linguistics, in at least two languages. This brute force approach could hardly overcome some the major problems in MT, such as ambiguity, structure mismatches, translation of idioms and etc.

structural mismatch occurs where two languages use the same construction for different purposes, or use different constructions for what appears to be the same purpose. Cases where the same structure is used for different purposes include the word ordering in English, and Japanese.

Satoo-san wa shyushoo ni erabaremashita.
(Satoo-hon TOP Prime Minister in was-elected)
Mr. Satoh was elected Prime Minister.

In this example, Japanese put the object in front of the verb and since Japanese allows subjects to be omitted freely, so one can say the equivalent of *elected Mr Satoh*, and thus avoid having to mention an AGENT. We can see that though different languages have their own way to construct a sentence, but they can convey the same meaning. So a good way to solve the structural mismatch problem would be to get rid of those differences in syntax by a standard semantic representation that both languages share. That divides the automated translation process into three steps: first the text of source language is transformed into the standard format of semantic representation with a semantic parser. And the semantic representation can be translated into the target language according to the semantic-syntax rules of the target language.

4.3 Question Answering

Question Answering has become an important and widely researched technique for information access because it can provide users with the exact information that they need rather than flood them with documents that they must wade through. Usually Information Retrieval techniques are used in QA. The information retrieval system will first retrieve relevant documents and information matching the question is extracted. Since it provides an answer instead of a document, it is more complex than an information retrieval system and such tasks require consideration of text meaning and how it could be matched to the question. There are multiple levels of understanding. The lowest levels are lexical level and syntactic level, which are both language dependent. An American needs to learn Russian lexicon and grammar to be able to understand questions in Russian. Once a question has been successfully mapped from its lexical-syntactic expression to a conceptualization, the subsequent interpretive processes are all language independent. It also suffers the problem that the same conceptual meaning may be represented by different lexical-syntactic expression.

Here is the different ways of asking for the same information.
When was Goerge Bush born?
What is Goerge Bushs birthday?

And here is the different ways of answering the questions.
Prescott and Dorothy Walker Bush gave birth to George in 1925.
George Bush was born in 1925.

To resolve these problems, a standard representation of the concept from an utterance is necessary, which is also the basis for higher level of understanding.

BIBLIOGRAPHY

[1] Understanding Semantics", Sebastian Lobner, Edward Arnold, London

[2] "Semantics", John I. Saeed, Blackwell Publishing 2003

[3] "Psychology of Language", Carroll. David W, Brooks/Cole Pub. c1999

[4] "Informal Lectures on Formal Semantics", Bach, Emmon W. Albany State University of New York Press, 1989

[5] "Natural Language Understanding", James. Allen, Pearson Addison Wesley

[6] "Handbook of Natural Language Processing", Dale, Robert, New York Marcel Dekker, Inc., 2000

[7] "Introduction to Automata Theory, Languages, and Computation", J.E. Hopcroft, Rajeev Motwani, J.D. Ullman, AddisonWesley Publishing Company

[8] "WordNet: An Electric Lexical Database", Christiane Fellbaum, MIT Press

[9] Fillmore, Charles J. (1976): Frame semantics and the nature of language, In Annals of the New York Academy of Sciences: Conference on the Origin and Development of Language and Speech, Volume 280 (pp. 20-32).

[10] "FrameNet: Theory and Practice", Christopher R. Johnson, Charles J. Fillmore, Miriam R. L. Petruck, Collin F. Baker, Michael Ellsworth, Josef Ruppenhofer, Esther J. Wood,

http://www.icsi.berkeley.edu/ framenet/book/book.html

[11] The Berkeley FrameNet project. Baker, Collin F., Fillmore, Charles J., and Lowe, John B. (1998), in Proceedings of the COLING-ACL, Montreal, Canada.

[12] "The Art of Prolog: Advanced Programming Techniques (Logic Programming), Leon Sterling, Ehud Shapiro, MIT Press

[13] "Natural Language Processing Techniques in Prolog", Patrick Blackburn, Kristina Striegnitz,

http://www.coli.uni-sb.de/ kris/nlp-with-prolog/html/

[14] "Perl 5 Complete" Peschko Ed, DeWolfe Michele, McGraw-Hill Professional