An Objective-Oriented Model for Semantic Analysis of Natural Languages

by

Abdul-Baqi Muhammad Sharaf-Al-Islam

A Thesis Presented to the

FACULTY OF THE COLLEGE OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

In

COMPUTER SCIENCE

January, 2001

An Object-Oriented Model for Semantic Analysis of Natural Languages

BY

ABDUL-BAQI MUHAMMAD SHARAF-AL-ISLAM

A Thesis Presented to the

DEANSHIP OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the

Requirements for the Degree of

MASTER OF SCIENCE

In

COMPUTER SCIENCE

KING FAHD UNIVERSITY
OF PETROLEUM & MINERALS
Dhahran, Saudi Arabia

JANUARY 2001

KING FAHAD UNIVERSITY FOR PETROLIUM AND MINERALS

DHAHRAN 31261, SAUDI ARABIA

DEANSHIP OF GRADUATE STUDIES

This thesis written by **Abdul-Baqi Muhammad Sharaf-Al-Islam**, under the direction of his Thesis advisor and approved by his Thesis Committee, has been presented to and approved by the Deanship of Graduate Studies, in partial fulfillment of the requirements for the degree of MASTER OF SCIENCE IN COMPUTER SCIENCE.

Thesis Committee

[illegible]
Dr. Mostafa Aref (Thesis Advisor)

[illegible]
Dr. Muhammad Al-Mulhem (Committee Member)

[illegible]
Dr. Moataz Ahmed (Committee Member)

Kanaan Faisal 24/01/2001
Department Chairman

[illegible]
Dean of Graduate Studies

عمادة الدراسات العليا
DEANSHIP OF GRADUATE STUDIES

٢٩/١٠/١٤٢١
24/1/2001
Date

Dedicated to....

My Mother

For her love and supplication

And to

My Father

For his support and encouragement

And to

My Wife

For her patience and Understanding

And to

My little Anas

For his smiles and joy

ACKNOWLEDGEMENT

All praise and glory to the almighty Allah – the exalted- who gave me courage and patience to carry out this work. Without His grace and mercy, this work would be impossible. Peace and blessings of Allah be upon the last Prophet Muhammad – Peace be upon him.

Acknowledgement is due to King Fahd University of Petroleum & Minerals for providing support for this work.My deep appreciation goes to my thesis advisor Dr. Mustafa Aref for his constant help, guidance and great hours of attention he devoted throughout the course of this research work. His priceless suggestions made this work interesting and learning for me.Thanks are due to my thesis committee members: Dr. Muhammad Al-Mulhem and Dr. Moataz Ahmed for their interest, invaluable cooperation and support. Thanks are also extended to Mr. Husni Al-Muhteseb for his valuable comments and suggestions.

Special thanks are due to my in-campus colleagues and friends for their help and encouragement. A few of them are Murtaza, Hasib, Arif and Rakib. My heartfelt thanks to my parents and other family members for their support and supplication. Special appreciation to my wife for her patience and understanding. Finally my thanks and love to my little Anas who added enjoyment at the middle of hardship.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

THESIS ABSTRACT

Name: Abdul-Baqi Muhammad Sharaf-al-Islam

Title: An Object-Oriented Model for Semantic Analysis of Natural Languages

Degree: MASTER OF SCIENCE

Major Field: Computer Science

Date of Degree: JANUARY 2001

Natural Language Processing (NLP) has many applications such as Database user interfaces, Machine Translation, Knowledge Acquisition and Report Abstraction. Natural Language Processing can be subdivided into various stages. Semantic Analysis is regarded to be the most important and the most challenging stage. This stage concentrates on converting the input sentence into an internal representation that reflects the meaning of the sentence. Any NLP system relies on a lexicon that contains information about language vocabulary.

The thesis proposes an Object-Oriented model for Semantic Analysis of Natural Languages. The model is named OSEMAN. OSEMAN assumes an object-oriented lexicon, where language words are stored as classes arranged in a semantic hierarchy. OSEMAN takes as input the stems of the sentence. Then it instantiates the stems from the sentence. These stems are combined to form an internal representation that reflects the meaning of the input sentence. These stems may combine in different ways that yields more than one internal representation for the same input sentence. Each representation is quantified by an overall weight that reflects the semantic consistency between the stems of the sentence.

King Fahd University of Petroleum & Minerals, Dhahran
January 2001

ملخص الرسالة

الاســـــم : عبد الباقي محمد شرف الإسلام

عنوان الرسالــة : نموذج للتحليل الدلالي للغات الطبيعية ذات خاصية التنسيق الذاتي

الدرجة العلميــة : ماجستير علوم

التخصـــص : علوم الحاسب الآلي

تاريخ التخصص : شوال ١٤٢١هـ

تلعب أنظمة معالجة اللغات الطبيعية دورا بارزا في كثير من التطبيقات النافعة، مثل واجهة قواعد البيانات، الترجمة الآلية، الإكتساب الآلي للمعرفة و التلخيص الآلي و غير ذلك. يمكن تقسيم أنظمة معالجة اللغات الطبيعية إلي عدة مراحل. يعتبر التحليل الدلالي للغات الطبيعية من أهم و أصعب المراحل. و تكمن مهمة هذه المرحلة في تحويل النص المعطى إلى بنية داخلية و التي تعكس المعاني و الدلالات الموجودة في النص المعطى. و يعتمد نظام معالجة اللغات علي قاموس يحوي معلومات لغوية عن المفردات.

تتقدم هذه الرسالة باقتراح نموذج للتحليل الدلالي للغات الطبيعية. و تتميز النموذج بتطبيق خاصية التنسيق الذاتي. و قد تم تسمية النموذج بـــ (OSEMAN). يفترض النموذج وجود قاموس ذات خاصية التنسيق الذاتي يقوم بتخزين المفردات على شكل فئات ضمن تصنيف دلالي. يتم إدخال النص المطلوب تحليله إلى النموذج. يقوم النموذج باستخراج الجذور من النص ثم ترتبط هذه الجذور مع بعضها لتكوين البنية الداخلية التي تعكس دلالات النص. و يمكن أن يكون هناك أكثر من بنية داخلية للنص الواحد. و في هذه الحالة يتم تقييم كل احتمال بقيمة تعطي دلالة على مدى قوة الارتباط بين الجذور لإعطاء دلالات النص.

جامعة الملك فهد للبترول و المعادن، الظهران
شوال ١٤٢١ هـ

CHAPTER 1

INTRODUCTION

Over the last decade, a tremendous advancement in computer speed and memory is observed. Similarly, advancements in computer networking enabled easy communication between remote machines. Now, the World Wide Web (WWW) allows millions of users from any part of the world to be exposed to a vast mass of information. One field that had stayed behind and could not cope with the rapid development of other fields and as a result caused a serious bottleneck in the way of efficient use of computers, is the field of Human-Computer interaction.

Human usually interact with computers through various programming languages. These languages allow users to perform different calculations and manipulation of data. However, these languages impose some difficulties. First, many programming languages fail to perform complex operations. Second, each programming language is strong in some aspects and weak in others. There is no language that is powerful in all aspects. This makes a heavy burden on the user in selecting the appropriate language and mastering it. If the user wants to write a program that needs features from different programming languages, then comes the problem of integrating different programming languages, which is not an easy task. Moreover, through programming languages, only highly professionals can interact with computers. These professionals represent only a small fraction of the total population of computer users, leaving the great majority behind.

On the other hand, if there exists an intelligent computer that understands natural languages, then a great obstacle is overcome. Human-Computer interaction will no more be restricted to specialized and professional programmers. With such intelligent computers at disposal, a user intending to travel to Riyadh, can input the following text:

> *"I want a reservation to Riyadh on economy class tomorrow at any time between 12:00 and 5:00 p.m."*

Getting this statement in pure natural language (English), the computer will process it and understand the request and will search the database for satisfied flights. Finally, the computer will present all satisfied flights to the user again in natural language.

The sub field of Artificial Intelligence that studies the possibility of developing intelligent computer capable of understanding natural languages is called **Natural Language Processing** or NLP for short. There are two motivating reasons for producing systems that are able to understand human languages:

1. Computers that understand languages can better search through and summarize the existing wealth of human knowledge.
2. Artifacts with the ability to make inference about concepts expressible in human languages will be able to make additions to the repository of human knowledge without restriction. In the near term, they will be able to double check the repository of consistency, validate new scientific claims, and suggest lines of research that have not yet been explored.

The high level goal is to produce a model that can infer underlying semantics given only surface realization. However, it is not an easy task for an intelligent computer to understand natural languages. It involves many stages and phases. An understanding system must first build a structure that reflects the syntax of the sentence. Next, the system must build an internal representation that reflects the meaning of the sentence. Finally, any references made in the sentence must me resolved and proper action is taken.

The most difficult stage is NLP is semantic analysis. This is true because natural languages are inherently ambiguous. A single statement can be interpreted in many different ways. The objective of this work is to come up with a model that transforms the textual English sentence into an internal representation that reflects the meaning of that sentence. The internal representation is a form that describes the semantic of the input sentence. There is no unified definition for that, and is usually dependent on applications that are going to use the internal representation. For example, in case of a database query system, the internal representation could be a query in SQL. When the input sentence is unambiguous, the model will produce a unique representation. In the case of ambiguity, the model will produce as many representations as the number of possible meanings of the sentence. In all cases, the model associates a weight with the internal representation(s). This weight reflects the syntactic and semantic consistency of the representations. Thus, in case of ambiguous sentences, internal representations are ranked according to their weight. To attain maximum flexibility, the model allows the user to impose constrains on how rigid the processing should be. Thus, constrains could be used to relax on syntactic errors or allowing metaphoric meaning of words. The usage of constrains will also be reflected on the weights assigned.

The thesis is organized in five chapters. Chapter 2 intends to give background information about Natural Language Processing (NLP) and Object Orientation methodology. Section 2.1 covers various issues related to Natural Language Processing. The need for NLP is discussed first, followed by discussing in brief the history of the evolution of NLP. Next, some works and trends in the NLP are presented. Various phases of NLP and the difficulties in the way of NLP concludes this section. Section 2.2 is devoted for semantic analysis; the phase on which this work is based. The need and motivation as well as the trends and difficulties of semantic analysis are discussed. Section 2.3 explains the concept of Object-Orientation in general. This chapter is concluded by section 2.4, where WordNet, a hierarchical lexical database that can be used in various NLP applications, is discussed.

Chapter 3 describes the model in detail. Section 3.1, discusses the object-oriented lexicon and its role in the model. It starts by finding the correspondence between natural language processing and object-orientation concepts. Section 3.2, gives an overview of the model. The internal mechanism of the model is explained by tracing all the phases and transactions carried on from the arrival of a sentence till a semantic representation is produced. The evaluation criterion is also discussed here. Section 3.3, discusses the robustness of the model and its ability to incorporate various difficulties in semantic analysis. Finally, section 3.4, presents the software development of the model using use-case driven approach. This approach is based on defining the whole system in terms of actors and use-cases.

Chapter 4 walks through the model by tracing four sentences as case studies. These sentences are studied in detail and various difficulties in their way of interpretation are emphasized along with describing the way in which the model overcame these difficulties. These sentences were chosen to reflect the diversity of natural languages. They are examples of ambiguous sentences possessing some elements of difficulty.

Chapter 5 represents the conclusion of the thesis. It summarizes the process of the model. It discusses how the model could be validated. It shows that the model is a valuable addition in the field of natural language processing. This chapter also mentions the advantages and limitations of the model. Finally, the chapter discusses the various ways the model could be improved in the future along with many applications where it could be used.

CHAPTER 2

BACKGROUND

This chapter intends to give background information about Natural Language Processing (NLP) and Object Orientation methodology. Section 2.1 covers various issues related to Natural Language Processing. The need for NLP is discussed first, followed by discussing in brief the history of the evolution of NLP. Next, some works and trends in the NLP are presented. Various phases of NLP and the difficulties in the way of NLP concludes this section. Section 2.2 is devoted for semantic analysis; the phase on which this work is based. The need and motivation as well as the trends and difficulties of semantic analysis are discussed. Section 2.3 explains the concept of Object-Orientation in general. This section shows that object orientation can be used in semantic analysis particularly and in natural language processing in general. This chapter is concluded by section 2.4, where WordNet, a hierarchical lexical database that can be used in various NLP applications, is discussed.

2.1 NATURAL LANGUAGE PROCESSING

Many different sciences have dealt with language processing from their perspective viewpoint. Computer science's perspective to language processing has given rise to a field in Artificial Intelligence called: Natural Languages Processing. Thus, NLP can be defined as the sub-domain of artificial intelligence concerned with the task of developing programs possessing some capability of 'understanding' a natural language in order to achieve some specific goal. Here 'understanding' refers to a transformation from one representation (*the input text*) to another (*internal representation*).

THE NEED FOR NATURAL LANGUAGE PROCESSING

If there exists a machine that understands natural languages, then the door will be open for hundreds of beneficial real-life applications. Here only few such important applications are presented.

1. ***Automatic Customer care through e-mail communication***: In technical domains, customer care is increasingly based on e-mail communication. Usually, customers pose their questions through e-mail, then a human expert is supposed to review the e-mails and give solutions in reply. This process is costly and time consuming. A machine that understands natural languages can make the process cost-effective and efficient. The system will read the message, understand it, and then will search for solution from given databases and finally send the solution to the sender. A similar approach based on shallow text processing and machine learning is described in [1].

2. ***Question Answering Systems***: These systems take a question as input and automatically provide one or more ranked answers to that question based on the set of material available to the system. Qanda is an example of question answering system [2]. A real world implementation of answering system is given in [35].
3. ***Summarization of Multiple Documents***: Usually a single source does not provide all information on a particular topic. Rather, scanning multiple related sources provide much more information. Summarizing all these information gives a valuable reference material on that topic. A machine that understands natural language can do the job in a much efficient way. MEAD is a multi-document summarizer which generates summaries using cluster centroids produced by a topic detection and tracking system [3]
4. There are many other useful applications for NLP. Just to name some: information retrieval [39], data extraction, machine translation [37], text categorization [38], story understanding [36], etc.

BRIEF HISTORY OF NATURAL LANGUAGE PROCESSING

NLP started out as people started building processing systems completely manually. That approach proved to be too difficult, or too cost-intensive. Recently, and inspired by successful machine learning systems, there has been a movement to create more NLP systems using inductive techniques from the machine learning community [4]. In what follows, different historical threads, which have given rise to the field of NLP is presented [5].

Foundation Insights: 1940- 1950

This period just after the World War II saw intensive work on two foundational paradigms: the automaton and probabilistic model. The automaton arose in the 1950s out of Turing's (1936) model of algorithmic computation. Kleen (1951) has contributed to finite automata and regular expression. Shannon (1948) applied probabilistic models of discrete Markov processing to automata for language. These early models led to the field of formal language theory. This period saw also a great advancement in the probabilistic algorithms. Shannon borrowed the concept of entropy from thermodynamics as a way of measuring the information content of a language, and performed the first measure of the entropy of English using probabilistic technique.

The Two Camps: 1957 – 1970

In the early 60s, the field of language processing has split into two paradigms: symbolic and stochastic. The symbolic paradigm took off from two lines of research. The first was the work of Chomsky and others on formal language theory and generative syntax, and the work of many linguistics and computer scientists on parsing algorithms. The second line of research was the new field of Artificial Intelligence (1956). The major focus of the new field was the work on reasoning and logic typified by Newell and Simon. At this point early natural language understanding systems were built.

The stochastic paradigm took hold mainly in departments of statistics and electrical engineering. By the late 1950s, Bayesian methods were beginning to be applied to the problem of optical character recognition. Beldsoe and Browning (1959) built a Bayesian system for text-recognition. 1960s also saw the rise of the first psychological models of human language processing based on transformational grammar, as well as the first on-line corpora: the Brown corpus of American English assembled at Brown University in 1963-64 which contains a collection of 1 million words from 500 written texts from different genres.

1970- 1983

The natural language understanding field took off during this period, beginning with Terry Winograd's SHRDLU system, which simulated a robot embedded in a world of toy blocks (1972). The program was able to accept natural language text commands. Roger Schank and his colleagues and students built a series of language understanding programs that focused on human conceptual knowledge such as scripts plans and goals, and human memory organization (1977). This work often used network-based semantics and began to incorporate Fillmore's notion of case roles into their representations. This period also saw a great advances in discourse

The Return of Empiricism: 1983 – 1993

In this period there was a rise of probabilistic models throughout speech and language processing, influenced strongly by the work at the IBM Thomas J. Watson Research center. These methods spread into part-of-speech tagging, parsing and disambiguation.

The Field Comes Together: 1994 – 2000

In this period probabilistic and data-driven methods had become quite standard. Increase in the speed and memory of computers had allowed commercial exploitation of a number of sub-areas of speech and language processing. Also the rise of the web emphasized the need for language based information retrieval and information extraction. [5].

TRENDS IN PROCESSING NATURAL LANGUAGES

Here the recent paradigms in handling NLP are presented along with some of the previous works done in this filed.

Statistical Approaches for NLP

Statistical approach, often called empiricist approach to NLP suggests that the complicated and extensive structure of language can be learnt by specifying an appropriate general language model, and then inducing the values of parameters by applying statistical, pattern recognition and machine learning methods to a large amount of language use. The goal is to discover language structure automatically. It is observed that language's usage changes gradually. The details of gradual change can only be made sense of by examining frequencies of use and being sensitive to varying strengths of relationships, and this type of modeling requires statistical observations. Thus, a major part of statistical NLP is deriving good probability estimates for unseen events.

For example, given a large corpus, we can easily estimate word types (different words) in the text and count of each type. Knowing this information the average occurrence of the words can be estimated. More useful information is provided by the **Zipf's law**, which inversely relates the frequency of a word to its position in the list. Zipf's law is useful as a rough description of the frequency distribution of words in human languages. Another useful law suggests that the number of meanings of a word is correlated with its frequency. Other laws of Zipf's include that there is an inverse relationship between the frequency of words and their length. Another law called **Power law** suggests that the probability of a word of length (n) being generated is $(26/27)^n(1/27)$. This result shows that there are 26 times more words of length (n+1) than length n, and that there is a constant ratio by which words of length (n) are more frequent than words of length (n+1). [6]

Agent Based Systems

Agent based model specifies agents that undertake certain objectives. The functionality of processing system is fulfilled when all these agents interact simultaneously. TALISMAN is a multi-agent system for NLP. It defines an agent called PRET for preprocessing, MORPH for morphological analysis, SEGM for splitting into clause, SYNT for syntactic analysis. Each agent contains all necessary components that are needed to perform its functionally. For example, the agent SYNT, contains verb dictionaries, grammars, indicator of the lexical structure and indicators of the grammatical structure and prepositions introducing adverbial sentences. This agent has access to date stored in a lexicon. This agent informs the other agents on agreement with gender, number and verb's choices or proposes solutions for ambiguous sequences. [7]

THE PHASES OF NLP

Natural Language Processing involves various stages. In each stage, concentration is given to achieve certain goals. Upon reaching the final stage, the understanding system should have created an internal representation that reflects the meaning of the input in natural language. Following subsections highlight the functionality of each phase.

Morphology

Morphology captures information about the shape and behavior of words in context. Most natural language systems begin the process at the word level. A word, however, is not atomic. For example, the word *unknowingly* can be further broken down, while the word *train* cannot. Taking the word *train*, it is known that the plural form is constructed by adding a trailing s at the end, and there are variations on the verb *to train* such as *training, trainer* and *trained.* The word in its simplest form is called **stem** (such as *train*) and all other morphemes attached to it is called **affixes**. (like s, ing, er, ed, etc.) Affixes may be further categorized by whether they are attached to the start or the end of the stem. They are called **prefixes** and **suffixes** respectively.

Words can be categorized into different syntactic types called **part-of-speech**. They represent a classification of words based on their use within a particular language. The major part-of-speech in English are: **Noun, pronoun, determiner, adjective, preposition, verb, adverb** and **conjunction.** A word may belong to more than one part-of-speech. *Train* might be a noun or a verb. There are certain rules for combining affixes to the root morphemes. Similarly, there are rules for possessive nouns. For adjectives and adverbs rules can be formulated for comparatives and superlatives. For verbs, the inflectional suffixes are used to alter the form of the verb. The root verb form is known as the infinitive. A good general rule for saying if a verb form is an infinitive is to ask if the verb form can be preceded by *to*. Thus, *to fly, to go, to hold* all indicated correct infinitive forms. The infinitive form provides no reference to a subject. Inflectional suffix to the infinitive form involves the third person singular (e.g., *he moves, he carries*). Infinitive form also provides no reference to tense. This information is provided by changes to the infinitive form to produce the past tense (e.g., *I **knew** him*). Knowing these morphological rules, only the stem forms of all words can be stored in the lexicon and write processes that will break a word into its morphemes. A device that performs such an analysis is called a **morphological analyzer**.

Syntax

Syntax is the study of how words fit together to form structures up to the level of a sentence. The most important part of sentence is the **verb**. Group of verbs working together (known as **verbal groups**) are often regarded as the most significant part of the structure of a sentence. In English, verbal groups are used to handle variations of tense, mood and modality. By **tense** of a verb it is meant: where the event occurs in time; past, present or future. Connected with tense is **aspect**. There is the **progressive aspect** e.g. *I am watching*, and the **perfect aspect**, e.g., *I have watched*. **Mood** of a verb means whether it is **active** or **passive** e.g., *I watch* (active) and *I am watched* (passive). **Modality** of a verbal group means indication of possibility or necessity or a degree of certainty. This is usually expressed by certain auxiliary verbs such as *can* and *may*. For example, *I can watch, I may watch*.

Words of a sentence fit together in certain patterns. A sentence can be regarded as being composed of a relatively small number of building blocks. A phrase is a constituent smaller than a simple sentence. Thus, the sentence: *He packed his bags* consists of two phrases: *He* and *packed his bags*, and that the latter phrase consists in turn of two phrases: *packed* and *his bags*. One could express the constituent structure of a sentence in a tree diagram as in the following figure.

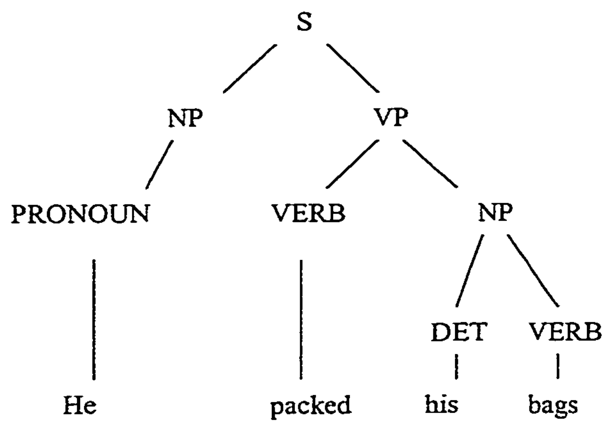


Figure 2.1 Parse tree for sentence *He packed his bags*

A **grammar** is a set of rules for showing the relationships between words. Context free grammars places a restriction on the formation of rules. Essentially this restriction is that there can only be one symbol on the left hand side of a rule. A context free grammar allows specifying complicated if-then type rules. A grammar that is powerful enough to be able to analyze all English sentences is an impossibly large and complex task. Below is a simple context-free grammar. [8]

```
1. S   ← NP VP
2. NP ← ART ADJ NOUN | ART NOUN | ADJ NOUN
3. VP ← AUX VERB NP | VERB NP
```

Figure 2.2 A simple context-free grammar

Given a grammar that describes the rules to build English sentences and a lexicon that contains the vocabulary of the language it should be possible to write a computer program to determine whether or not any given text is constructed according to the rules of grammar. If a sentence is grammatical then the program should be able to describe its structure. If a sentence is ambiguous, then the program should be able to describe all its possible structures. A program that performs such functions is called a parser.

Semantics

Semantic analysis concentrates on the meaning of a sentence. A sentence can be syntactically correct but semantically inconsistent. For example, *Ahmad is eating the door*. Section 2.2 is entirely dedicated for semantic analysis.

Discourse

By the completion of the previous phases, we have information about the structure of a sentence as well as the meaning of its words. However, it is still needed to resolve the pronoun references in the sentence and thus, map the entities mentioned in the sentence to the real world. For example, consider the sentence, '*Bill had a red balloon. John wanted it.*' It must be determined that the pronoun '*it*' refers to '*balloon*'. Also it must be known who '*Bill*' is. The objective of discourse analysis is to resolve the reference sited in the sentences. If the reference is made by a pronoun, then the task is fairly easy; we need to refer back to the nearest entity in most of the cases. This kind of reference resolution is called, **anaphoric reference resolution**. However, references in a sentence might be more complex and not just denoted by a pronoun. Consider the sentence, '*Sue opened the book she just bought. The title page was torn.*' The phrase '*the title page* ' should be recognized as being part of the book that was just bought.

Pragmatics

After passing through the previous phases, a complete description is present in the terms provided by our knowledge base. The final step toward effective understanding is to decide what to do as a result. For example, given the sentence, '*Do you know what time it is*?' should be interpreted as a request to be told the time, and not just a '*yes/no*' question. This sentence suggests that in natural languages, something is said of which literal meaning is not intended. For example, '*watch your steps*' is actually meant to *be careful* and not just the literal meaning of *watching one's steps*.

DIFFICULTIES AND CHALLENGES IN NLP

There are many difficulties in the way of proper understanding of Natural languages. The most challenging one is ambiguity. Hence, the focus will be on ambiguity and trends in resolving it.

Ambiguity

Ambiguity is the biggest challenge in the way of proper understanding of natural languages. In a natural language expression, ambiguity occurs when there is more than one interpretation. Consider the sentence: *I made her duck.* This sentence can have the following interpretation:

1. *I cooked waterfowl for her.*
2. *I cooked waterfowl belonging to her.*
3. *I created the (plaster?) duck she owns.*
4. *I caused her to quickly lower her head or body.*
5. *I waved my magic wand and turned her into a waterfowl.*

From the above example it is evident that ambiguity occurs in all levels. It can be in the word level (as with the word *duck*) or it can be in the sentence level. There are two main types of ambiguity, **lexical** and **structural**.

Lexical Ambiguity

Lexical ambiguity occurs when one word has more than one meaning. Nouns like *chip, pen* and *suit*, verbs like *call, draw* and *run* and adjectives like *deep dry* and *hard*, are all lexically ambiguous. After through analysis, some sources for lexical ambiguity can be identified. When studying natural languages some regularity can be observed. For example, one morphological regularity says that, in English, adding the suffix –er to a verb yields a noun that means 'a person who performs the act denoted by the verb'. Thus, *play* yields *player*. The notion of ambiguity comes when parsing an input more than one rule can be applied. For example, if there is a word ending in the suffix –er, it could be a noun of the type described above, but it could also be a comparative adjective. For example, the word *thinner* could be a comparative adjective (*this man is thinner that that man*) or a noun (*the paint thinner*).

Another source of lexical ambiguity arises when one word has more than one sense, as with the noun *bank* which could mean the *bank of a river* or *the monetary transaction place*. Sometimes a verb has a transitive sense and another intransitive sense as with *burn, fly, make* and *walk*.

To disambiguate lexical ambiguity, **part-of-speech tagging** is an effective way. Part-of-speech (POS) tagging is the process of resolving lexical syntactic ambiguity and annotating each word in a sentence with a POS label or tag. There have been many attempts for POS. TAGGIT is a program created by [9] for tagging the Brown corpus. The process is divided into two stages. Initial tag selection identifies all possible tags for a word, and tag disambiguation chooses the most appropriate tag. Tag disambiguation is based on manually created heuristic rules, which use a word's local context to determine the correct POS. Frequency-based tagging program called Constituent Likelihood Automatic Word tagging System or CLAWS is described in [10]. An alternative approach that is based on a Hidden Markov Model is described in [11]. An empirical approach for POS tagging that is based on loglinear model is given in [12].

Structural Ambiguity

The syntactic structure of a sentence indicates how the part of the sentence should be combined during interpretation. When more than one syntactic structure could be assigned to a given sentence, the sentence is structurally ambiguous.

Following examples illustrates the notion of structural ambiguity.

1. *Bengali history teacher*. (The nationality of the history teacher is Bengali, OR the teacher is specialized on Bengali History)
2. *Short men and women*. (Short men and short women, OR men are short but women are of any height)
3. *The girl hit the boy with a book*. (The girl hit the boy who was carrying a book, OR, The girl used a book to hit the boy)

4. *Visiting relatives can be boring.* (Going to the see the relatives is boring, OR, When relatives come to visit us, it becomes boring)
5. *The chicken is ready to eat.* (The fried chicken is well prepared to be eaten, OR, the hungry chicken is ready to have some meal).
6. *Ali knows a richer man than Ahmad.* (Ali knows a man who is richer than Ahmad, OR, Ali knows a man who is richer than any man Ahmad knows)
7. *Adel loves his mother and so does Esam.* (Adel loves his mother and Esam loves his mother, OR, Adel loves his mother and Esam also loves Adel's mother.)
8. *Everybody loves somebody.* (for each person there is somebody this person loves, OR, there is somebody that everybody loves).

Prepositional Phrases (PP) are one of the most frequent sources of structural ambiguity. English PPs typically occur towards the end of the sentence, which allows them to attach to most of the preceding constituents of the sentence. Most studies have limited their attention to the problem of a PP with only two possible attachment sites, a verb and a noun phrase. Many PPs are not limited to just two possible attachment sites that differ in syntactic category. To illustrate this point, here are some examples showing some different possible attachment sites for English PPs.

- Right Association: Attachment to most recent NP

  Example: '*Clean the surface of the tube.*'

  The PP *the tube* is attached to most recent NP: "*the surface*"

- Minimal Attachment: Attachment to NP

Example: *Check the beam focus with the supplied instrument*

Here the PP *the supplied instrument* is attached to the VP *check.*

- Attachment to a higher NP:

  A PP can attach to a higher NP, i.e., an NP that dominates the most recent NP in the parse tree.

  Example: *Record the signal strength at the wire under normal load*

  Here the last PP *normal load* is attached to the higher NP *the signal.*

- Attachment to an Adjective Phrase:

  Example: *Temperatures above 80 degrees can be harmful to the unit*

  Here the PP *the unit* is attached to the adjective *harmful.*

There has been several approaches to the resolution of structural ambiguity.

- *Syntactic Approaches*: these approaches use structural properties of the parse tree to choose a particular parse. A number of heuristics for syntactic ambiguity is given in [13]. Using these strategy garden-path effect can be eliminated. In a garden-path sentence (e.g., *The horse raced past the barn fell)* the reader begins to interpret the sentence in accordance with canonical sentoid strategy (which shows that the first N..V..N clause is the main clause unless the verb is marked as subordinate), but the correct interpretation turns out to be a different interpretation, and the reader needs to make an effort to re-interpret the sentence.
- *Semantic approaches*: not all cases of syntactic ambiguity can be resolved with reference to syntactic properties of the sentence alone, so there has been a significant

amount of work on semantic features for disambiguation. The selection restrictions of [14] are an early example. They assigned semantic binary features, such as ANIMATE which provides constraints on syntactic roles such as arguments of verbs. For example, *Mary drank the glass*, where *glass* is ambiguous. In order to determine the correct sense of the noun glasses (*drinking vessel* or *spectacles*) some semantic inference has to be performed. Thus, in the limit, ambiguity resolution requires semantic knowledge.

- *Statistical approach*: A disambiguation method based on co-occurrence data is described in [15]. The method employs a probability model, which naturally represents co-occurrence patterns over word pairs and makes use of an efficient estimation algorithm based on the Minimum Description Length (MDL) principle. The method is able to disambiguate compound nouns (e.g., *data base systems*) and PP-attachment like (*see boy with telescope*). To give more elaboration, with *see boy with telescope*, it is needed to determine to which of *see* or *boy* the phrase *with telescope* should be attached. To handle this, the method compares the probabilities: `P(telescope|see)` and `P(telescope|boy)`, and will pick the verb see to be the appropriate candidate.

There have also been some other works in resolving the PP attachment problem. The notion of lexical preferences for ambiguity resolution was introduced in [17], and a method for learning lexical association strengths for PP attachment from a text corpus was described in [16]. A series of experiments that included the prepositional objects was carried by [18]. The nominal synonym sets from WordNet were used to provide the word classes. Both the attachment site and prepositional object are placed into a semantic or conceptual class, and the relative strengths of the conceptual association between classes via the preposition were estimated from a text corpus. An empirical method bases on loglinear model for PP attachment is given in [12].

2.2 SEMANTIC ANALYSIS

Semantic analysis is one of the most important phases of Natural Language processing. Without a proper semantic analyzer, An NLP system will be incomplete. This section discusses various issues related to this topic.

The objective of semantic analysis is to produce a representation of the meaning of the sentence. Because understanding is a mapping process, it must be determined first the language into which mapping is performed. In the case of any high-level programming language, the meaning of the program is the machine code produced by the compiler. Unfortunately, there is no single definitive language in which all sentence meanings can be described. Rather, this target language depends on what is to be done with the meaning once they are constructed. For example, if the system is an interface to a database, then the SQL would be a good candidate for the target language. Although the main purpose of semantic processing is the creation of a target language representation of a sentence's meaning, there is another important role that it plays. It imposes constraints on the representations that can be constructed. Semantic processing can impose constraints because it has access to knowledge about what makes sense in the world.

The first step in any semantic processing system is to look up the individual words in a dictionary and extract their meanings. Unfortunately, many words have several meanings, and it may not be possible to choose the correct one just by looking at the word itself. One way to get around this problem is associating with each word in the lexicon, information about the contexts in which each of the word's senses may appear. This extra information is called markers. Common markers could be LOCATION, ANIMATE, PHYSICAL, etc. To solve the lexical disambiguation problem completely, it is necessary to introduce more and more finely grained semantic markers. As the number of such markers grows, the size of the lexicon becomes unmanageable. In addition, each new entry into the lexicon may require that a new marker be added to each of the existing entries.

The meaning of a sentence depends on the meaning of the individual words in the sentence and how these words are related to each other. Several approaches to the problem of creating a semantic representation of a sentence have been developed. **Semantic grammar** is an approach that combines syntactic, semantic, and pragmatic knowledge into a single set of rules in the form of a grammar. The result of parsing with such a grammar is a semantic, rather than just a syntactic, description of a sentence. **Case grammar** is another approach, in which the structure that is built by the parser contains some semantic information, although further interpretation may also be necessary. In **conceptual parsing** the syntactic and semantic knowledge are combined into a single interpretation system that is driven by the semantic knowledge. [19]

DIFFICULTIES

There are many difficulties arise when we attempt to implement a semantic analyzer. The following two are more significant:

Internal representation: The semantic analyzer transforms the input sentence (in form of stems) into an internal representation that captures the meaning of the input sentence. Some of the issues regarding internal representation is listed:

a. ***Way of presentation***: The first thing that must be fixed is to agree upon the paradigm that should be used in presenting the internal representation. [8] describes some of the common paradigms to represent the semantics. Among them are: *Logical forms* (LF) and *Semantic Networks*. In our work we advocate the use of Object-Oriented paradigm.

b. ***Ability to Capture the meaning***: The most important characteristic of the internal representation is it's ability to reflect the meaning of the sentence. A good system for the representation of knowledge in a particular domain should process the following four properties: [19]

   1. **Representational Adequency**: The ability to represent all of the kinds of knowledge that are needed in that domain.
   2. **Inferential Adequency**: The ability to manipulate the representational structures in such a way as to derive new structures corresponding to new knowledge inferred from old.
   3. **Inferential Efficiency**: The ability to incorporate into the knowledge structure additional information that can be used to focus the attention of the inference mechanisms in the most promising directions.

4. **Acquisitional Efficiency**: The ability to acquire new information easily.

Transformation: Mapping a set of stems produced by the morphological analysis into an internal representation. The input to this algorithm will be a set of stems, which will be considered as the building blocks. The algorithm will then relate these blocks in a way that reflects the meaning, which is nothing but the internal representation. There are few issues need to be studied in the transformation process.

1) ***Type of mapping***: There should be an algorithm by which the set of inputs will be mapped to the set of internal representations in a many-to-many fashion. There were few strategies discussed in the literature for the transformation process [8]. Interpretation rules, semantic grammars and Semantic interpretation based on preferences are some of these strategies. The difficulty lies in the fact that natural languages consist of a wide range of vocabulary, and each word has many senses in most cases. The proper sense must be recognized from the context. Coming up with a rigid algorithm to pick up the appropriate sense of the word from the context is not trivial.

2) ***Ambiguity***: Natural Languages are inherently ambiguous. Consider the sentences:

Time flies like an arrow. [12]

Each words of this sentence has more than one sense. 'Time' can be a verb and also a noun. Similarly 'fly' can be either a verb or noun. 'like' can be thought of as 'similar to' or as 'found of'. Some of the possible interpretations are given below:

1) Time passes along in the same manner as an arrow gliding through space.
2) I order you to take timing measurements on flies, in the same manner as you would time an arrow.

3) Fruit flies like to feast on a banana; in contrast, the species of flies known as "time flies" like an arrow.

In the transformation phase all of these possibilities must be considered and hence, several internal representations should be presented, in hope that in further processing by discourse analysis this ambiguity will be resolved.

4) ***Difficulty in interpreting some language constructs***: There are some language constructs that deserve special attention while designing transformation algorithm. Examples of these language constructs are:

   (1) **Noun sequences:** These are sequence of nouns that come next to each other. The first noun is called the *modifier* and the second is the *head*. For example; *plum sauce* and *knowledge representation*. The challenge lies in the fact that there are different semantic relations that relates the first noun with the second. In *Plum sauce*, the modifier *plum* specifies the ingredient of the head: *sauce*. But, in the second NS, the modifier *knowledge* specifies the object of representation.

   (2) **Verbal polysemy :** This refers to the fact that a verb will have different senses that depends on the context where it is used. For example, the verb *take* can mean *buy* in the sentence: '*He took a minority stake in the venture*', or mean *spend* in the sentence : '*That process is likely to take time*', or even mean *obtain* in the sentence: '*People are rushing to take products of Microsoft*'.

   (3) **Nominalization:** It is a noun phrase that has a systematic correspondence with a clause structure, where the head noun of nominalization phrase is related morphologically to either a verb (deverbal noun) or adjective (deadjectival noun). For example, the noun "*decoration*" has many senses among them is the nominalization sense, which means the act or process of decoration.

5) ***Using the Lexicon***: Lexicon is a vocabulary data bank that contains the language words and their linguistic information. While constructing the internal representation there must be a mechanism to link with the lexicon and extract information from it.

PREVIOUS WORK

Here some of the works done in the field of semantic analysis is presented.

Approaches based on semantic roles

An approach based on case frames with a head and multiple slots is described in [20]. For example, we can represent the sentence *I ate ice cream* as `(eat (`arg_1` I) (`arg_2` ics-cream))` where the head is '*eat*', arg_1 slot represents the subject and arg_2 slot represents the direct object. Furthermore, any member of the word class <animal> can be incorporated as the case frame pattern for the verb 'eat' in arg1 slot and a member of the word class <food> can be the value of the arg2 slot. Classically, case frame patterns are represented by 'selection restrictions' which is a discrete representation of semantic features, but in this approach it is represented continuously because a word can be the value of a slot to a certain probabilistic degree as is suggested by the following list

1. *Adel drank some water*
2. *Adel drank some gasoline*
3. *Adel drank some pencils*
4. *Adel drank some sadness.*

Semantic interpretation of nominalization is addressed in [21]. Interpretation of nominalization involves three tasks: deciding whether the nominalization is being used in a verbal or non-verbal sense, disambiguating the nominalized verb when a verbal sense in used and determining the fillers of the thematic roles of the verbal concept or predicate of the nominalization. Determining the verb meaning is done by rules that use the types of syntactic relations built by the parser and the semantic categories of the nouns in them. Once the verbal concept of the nominalization has been determined, the next step is to determine which of the other constituents of the NP fill the thematic roles of the verbal concept and what those roles are. This is done by the selection restrictions associated with the roles. An algorithm for automatic interpretation of noun sequence is given in [22]. The system uses broad coverage semantic information, which has been acquired automatically by analyzing the definitions in an on-line dictionary.

Machine Learning Approaches

A corpus-based approach for building semantic lexicon is described in [23]. The input to the system is a text corpus and a small set of seed words for a category. The output is a ranked list of words that are associated with the category. A user then reviews the top-ranked words and decided which ones should be entered in the semantic lexicon. The algorithm to select words that belong to the same category of the seed words uses simple statistical and bootstrapping mechanism. The work is based on the observation that category members are often surrounded by other category members in text.

A different machine learning technique was adopted by [24] for noun phrase recognition. Classifiers were used for this purpose. Different classifiers were used by using different representations of data. By combining the results with voting techniques, the performance was improved.

An automatic recognition algorithm for verbal polysemy is given in [25]. Their work is based on clustering methods, which decides first when a verb has to be split into two hypothetical verbs based on the semantic compactness of a group of verbs. Then, the algorithm produces a set of semantic clusters.

Logic-based Approaches

A logic-like language for semantic representation of English questions is developed by [26]. The system was named as TOP (Temporal Operator). TOP was designed to support the systematic representation of English time related semantics. TOP atomic formulae are constructed by applying predicate symbols to constants and variable. More complex formulae are constructed using conjunctions and temporal operators. For example, the question: Was tank 5 empty on 1/1/98 ? can be represented using TOP as:

$$At\ [1/1/98,\ Past[e^v,\ empty(tank5)]]$$

The 'v' suffix marks variable, and free variables are treated as quantified by an implicit existential quantifier with scope over the entire formulae. The verb in the above sentence introduces a Past operator which requires empty (tank5) to bave been empty at some past time e^v and the 'at 1/1/98' adverbial introduces an At operator, which requires that past time to fall within 1/1/98.

2.3 OBJECT-ORIENTATION CONCEPTS

In object-orientation, a system is modeled as number of objects that interact with each other. The surroundings, for instance, consist of objects, such as people, trees, cars, towns and houses that are in some way related to each other. A model, which is designed using an object-oriented technology, is often easy to understand, as it can be directly related to reality. Thus, with such a design method, only a small semantic gap will exist between reality and the model. The concept of object-orientation will be introduced independent of the programming languages.

ABSTRACTION

The most prominent feature of object-orientation is abstraction. Abstraction is achieved through the concept of Objects and Classes.

Objects

The first and the most important concept is the concept of object. Object is an entity able to save a state (information) and which offers a number of operations (behavior) to either examine or affect this state. Objects usually correspond to real-life entity objects, such as a person or a car for example. Taking a person as an object, behavior and information is attached to him. Examples of information are age, address, male/female and so on. To access or be able to affect this information, for each object, a set of operations must be defined that can affect or read the saved information. It is possible also to define operations that perhaps need not affect any internal information, but only perform a behavior (for example, walk, jump, eat). The only part of the object seen is its operations; the inside is hidden.

If the inside of an object is exposed, then both its information structure and how its operations work will be seen. Attributes show the parts that the object consists of and how the behavior for the operations is defined. The interaction in an object-oriented model are created by means of objects sending messages to other objects. A message, which is received by an object, causes an operation to be performed in the receiving object. All information in an object-oriented system is stored within its objects and can only be manipulated when the objects are ordered to perform operations. The behavior and information are encapsulated in the object. The only way to affect the object is to perform operations on it. Objects thus support the concept of information hiding, that is, they hide their internal structure from their surroundings. Encapsulation means that all that is seen of an object is its interface, namely the operations we can perform on the object.

Classes

A class is a definition, template or a mold to enable the creation of new objects and is, therefore, a description of the common characteristics of several objects. The objects comprising a certain class have this template in common. For example, *Ahmad*, *Adel* and *Asma* are all people who have similar behavior and information structure, thus, they all belong to the same class: *person*.

In object-oriented systems, each object belongs to a class. An object that belongs to a certain class is called an instance of that class. Thus, *Ahmad, Adel* and *Asma* are all instances of the class person. The behavior of the instance and its attributes are defined by its class. In this way, the class person will contain the attributes (age for example) and operations to examine or modify this age. When an instance of the class *person* is created (for example, *Ahmad*), it is needed to store Ahmad's age in this instance. If later, it became necessary to distinguish between men and women in some features, for example the way they talk, then it can be done easily by creating two classes, one for Male and one for Female.

POLYMORPHISM

Polymorphism is one of the powerful features of object-orientation. It enables efficient interaction between objects of different classes. When an object sends a message to another object and the sending object does not have to be aware of which class the receiving instance belongs to, we say that we have polymorphism. To explain more, through polymorphism an object can send the same message to two different objects of two different classes (sending a message is nothing but invoking a method in the receiving object). Thus, the method is implemented differently in different objects of different classes. For example, consider a community where there are objects of male and female classes. An object called *Ahmad* might send a message called walk to three of his neighbors: *Adel*, *Ali* and *Asma*. Though the receiving objects belong to two different classes (*Adel* and *Ali* belong to class Male, whereas; *Asma* belongs to class Female), all of them have the method walk, but with different implementation. It is the receiver of the message that determines how a stimulus will be interpreted, not the sender. The sender need only to know that another instance can perform a certain behavior, not, which classes the object, belongs to. This is an extremely strong tool for allowing us to develop flexible systems.

INHERITANCE

When describing classes, it is soon noticed that many classes have common characteristics (behavior and attributes). For instance, when comparing the classes Male and Female, it is evident that they are very similar to each other. The similarities can be shared between the classes by extracting them and placing them in a separate class Person. In Person, everything that is common to Male and Female are included. In this way several classes can share common characteristics. In this way common characteristics are collected in one specific class and all other classes sharing these features are allowed to inherit this class. Only characteristics that are specific to the inheriting class are included in the class itself. Inheritance allows reuse of common description. Other advantages can also be gained through inheritance. If some changes are needed then it could easily be done in the parent class only, which will be automatically reflected in the children (inherited) classes. Hence inheritance is very useful for easy modification of models. Moreover, through inheritance repetition of common information is avoided. Inheritance is thus also useful for avoiding redundancy, leading to smaller models that are easier to understand. The ease of modification does not only occur inside class descriptions. Adding new classes can be easily done by just describing changes to existing classes. By means of extracting and sharing common characteristics, it is possible to build a chain of hierarchies where classes descend from general to more specific. Similarly, adding new classes can be incorporated easily by making it inherit a class that already offers some of the operations and information structure required for the new class.

PREVIOUS WORKS

There were only few attempts in exploiting features of object-orientation features in the field of Natural language processing. This was a motivation to work in this expanding field. Here some of the previous works are mentioned.

A general framework for Object-Oriented Natural Language Processing is presented in [27]. Figure 2.3 depicts the model proposed. An Object-Oriented algorithm for morphological analysis is described in [28]. A proposal for an object-oriented approach for semantic analysis is presented in [29].

A denotational semantic is described in [30] that utilizes the power of both object-oriented programming and Object-centered knowledge representation and constructs a bridge between the two. The result is a framework that can be used in natural language understanding.

A bilingual semantic knowledge representation is described in [31]. The concept adopted in this paper is much similar to the concept adopted here. A vocabulary word is considered as an object with attributes and methods. The meanings will be language independent. If an object represents the meaning of a word, additional attributes are attached to that object describing the realization of that meaning in different languages.

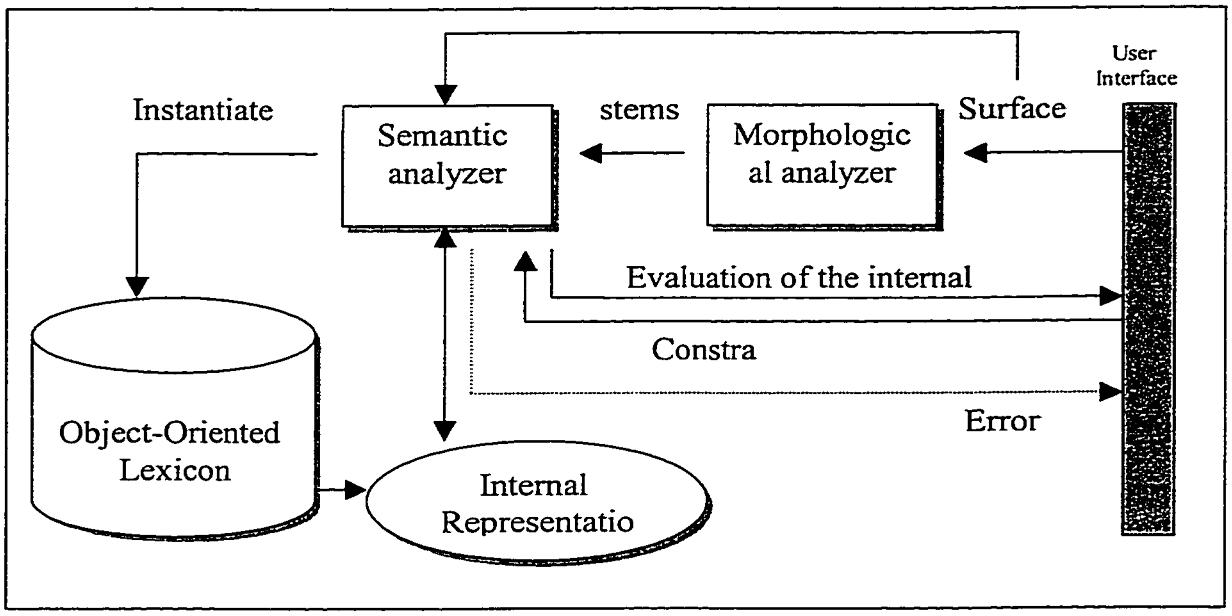


Figure 2.3 An Object-Oriented model proposed in [27]

An object oriented tool that converts English military operations orders or similar planning discourse into C++ and Java objects, with appropriate attributes, methods and relationships is described in [32]. The system handles only command specific sentences. The process first passes through the conventional language understanding stages. The information extracted from these stages is then converted into C++ of Java classes.

Logons - an object oriented natural language generation system- is described in [33]. The input to the system is a sentence-object, which contains semantic parser, required word constraints, and a syntactical type of the desired sentence. The sentence-object gives us the ability to generate sentences that express a desired intention or meaning. The more knowledge expressed in the sentence-object, the less randomness results in the sentence. The system has the ability to select random words that would fit the sentence semantically, If the sentence-object does not contain all the required word constraints. This will guarantee the generation of complete and semantically correct sentences.

2.4 WORDNET: A LEXICAL DATABASE

In 1985 a group of psychologists and linguists at Princeton University undertook to develop a lexical database. The initial idea was to provide an aid to use in searching dictionaries conceptually, rather than merely alphabetically. WordNet is the result. WordNet presently contains approximately 95600 different word forms organized into some 70,100 word meanings, or sets of synonyms. The most obvious difference between WordNet and a standard dictionary is that WordNet divides the lexicon into four categories: nouns, verbs, adjectives and adverbs. Nouns are organized in lexical memory as topical hierarchies, verbs are organized by a variety of entailment relations, and adjectives and adverbs are organized as N-dimensional hyperspaces. The reason for using WordNet in this thesis, as is the case in many NLP programs, is because WordNet attempts to organize lexical information in terms of word meanings, rather than word forms.

The basic design of WordNet is based on lexical semantics which relates word meanings to word forms. This can be depicted by the following lexical Matrix.

Word Meanings	Word Forms			
	F_1	F_2	F_3...........	F_n
M_1	$E_{1,1}$	$E_{1,2}$		
M_2		$E_{2,2}$		
M_3			$E_{3,3}$	
.				
.				
.				
.				
M_m				$E_{m,n}$

Table 2.1 Illustration of the concept of a Lexical Matrix

In the matrix notice that F1 and F2 are synonyms and F2 is polysemous. In WordNet word meanings can be represented by simply listing the word forms that can be used to express it: {F1,F2,....}. this is called synsets. For example, someone who knows that a *board* can signify either *a piece of lumber* or *a group of people assembled for some purpose* will be able to pick out the intended sense with no more help than *plank* or *committee*. The synonym sets, {*board, plank*} and {*board, committee*} can serve as unambiguous designators of these two meanings of *board*. These synonym sets do not explain what the concepts are; they merely signify that the concept exists. WordNet is organized by semantic relations. Since a semantic relation is a relation between meanings, and since meanings can be represented by synsets, it is natural to think of semantic relations as pointers between synsets.

NOUNS IN WORDNET

Definitions of common nouns typically give a superordinate term plus distinguishing features; that information provides the basis for organizing noun files in WordNet. For example, the noun tree can be defined in terms of its superordinate plant plus distinguishing features like: large, woody, perennial, and has distinct trunk. The superordinate relation (hyponymy) generates a hierarchical semantic organization that is duplicated in the noun files by the use of labeled pointers between sets of synonyms (synsets). The hierarchy is limited in depth, seldom exceeding more than a dozen levels. Distinguishing features are entered in such a way as to create a lexical inheritance system, a system in which each word inherits the distinguishing features of all its superordinates. Although the overall structure of noun hierarchies is generated by the hyponymy relation, details are given by the features that distinguish one concept from another. For example, a canary is a bird that is small, colorful, sings, and flies, so not only must canary be entered as a hyponym of bird, but the attributes of small size and bright color must also be included, as well as the activities of singing and flying. Moreover, canary must inherit from bird the fact that it has a beak and wings with feathers. In order to make all of this information available when canary is activated, it must be possible to associate canary appropriately with at least three different kinds of distinguishing features:

1. Attributes: *small, yellow*
2. Parts: *beak, wings*
3. Functions: *sing, fly*

The coverage of English nouns in WordNet is done through partitioning them into topical groups semantically. Twenty-five files were created and each partition is started with a unique beginner. [34]

ADJECTIVES IN WORDNET

Adjectives modify nouns. The lexical organization of adjectives is unique to them, and differs from that of the other major syntactic categories, noun and verb. WordNet presently contains approximately 19,500 adjective word forms, organized into approximately 10,000 word meanings (synsets). WordNet contains descriptive adjectives and relational adjectives.

Descriptive adjective is one that ascribes a value of an attribute to a noun. The semantic organization of descriptive adjectives is entirely different from that of nouns. The basic semantic relation among descriptive adjectives is antonymy. The importance of antonymy in the organization of descriptive adjectives is understandable when it is recognized that the function of these adjectives is to express values of attributes, and that nearly all attributes are bipolar. Antonymous adjectives express opposing values of an attribute. For example, the antonym of *heavy* is *light*, which express a value at the opposite pole of the WEIGHT attribute.

Relational adjectives mean something like 'of, relating to, or associated with' some noun, and they play a role similar to that of a modifying noun. For example, *fraternal*, as in *fraternal twins* related to *brother*, and *dental*, as in *dental hygiene*, is related to *tooth*. Relational adjectives differ from descriptive adjectives in that they do not relate to an attribute: there is no scale of criminality or musicality on which the adjectives in criminal law and musical training express a value. Since relational adjectives do not have antonyms, they cannot be incorporated into the clusters that characterize descriptive adjectives. As a result, WordNet maintains a separate file of relational adjectives with pointers to the corresponding nouns. Some 1,700 relational adjective synsets containing over 3,000 individual lexemes are currently included in WordNet. Each synset consists of one or more relational adjectives, followed by a pointer to the appropriate noun. For example, the entry *{stellar, astral, sidereal, noun.object:star}* indicates that stellar, astral, sidereal relate to the noun star.

English color terms are exceptional in several ways. They can serve as either nouns or adjectives, yet they are not nominal adjectives: they can be graded, nominalized, and conjoined with other descriptive adjectives. The organization of color terms is given by the dimensions of color perception: lightness, hue, and saturation, with define the well known color solid. In WordNet, hues are coded as similar to colored, and the shades of gray from white to black are coded as similar to gray, which is in a tripartite cluster with white and black, providing for a graded continuum. [34]

VERBS IN WORDNET

Verbs are arguably the most important lexical and syntactic category of a language. All English sentences must contain at least one verb. Currently, WordNet contains over 21,000 verb word forms and approximately 8,400 word meanings (synsets). Included are phrasal verbs like look up and fall back. Verbs are divided into 15 files, largely on the basis of semantic criteria. All but one of these files correspond to what linguists have called semantic domains: verbs of bodily care and functions, change, cognition, communication, competition, consumption, contact, creation, emotion, motion, perception, possession, social interaction, and weather verbs. Virtually all the verbs in these files denote events or actions. Another file contains verbs referring to states, such as *suffice, belong,* and *resemble*, that could not be integrated into the other files.

The principle of lexical inheritance can be said to underlie the semantic relations between nouns, and bipolar oppositions serve to organize the adjectives. Similarly, the different relations that organize the verbs can be cast in terms of one overarching principle, **lexical entailment**. For example, *snore* lexically entails *sleep* because the sentence *He is snoring* entails *He is sleeping*; the second sentence necessarily holds if the first one does.

SUMMARY

This chapter gave a general overview of Natural Language Processing. That included a brief history of the development of NLP. Different phases of NLP were described with detail discussion of semantic analysis. Ambiguity, as a challenge in the way of proper NLP, is covered in detail. Object-Orientation concepts were discussed, as the model advocates using this paradigm. Attempts were made to relate various concepts of Object-Orientation to NLP. The chapter was concluded with a general overview of WordNet, a lexical database. The hierarchy described in WordNet can be used in designing an Object-Oriented lexicon.

CHAPTER 3

AN OBJECT ORIENTED MODEL FOR SEMANTIC ANALYSIS (OSEMAN)

Both semantic analysis and Object-Orientation are discussed in the previous chapter. Different objectives, requirements and challenges of semantic analysis were mentioned. This chapter describes a model that uses object-oriented principles in performing the function of a semantic analyzer. The model is named OSEMAN (**O**bject-Oriented **SEM**antic **AN**alyzer). OSEMAN assumes the presence of an object-oriented lexicon where language words are represented as objects and their semantic and linguistic information are encoded in terms of attributes and behaviors. Furthermore, similar words are related through inheritance. Therefore, given a sentence consisting of words, OSEMAN will instantiate these words from the lexicon and will let these word objects search for their suitable partners to come up with a semantically consistent structure that reflects the meaning of the sentence. At the end OSEMAN gives as output some weighted representations of the input sentence that reflect the most likely meanings of the sentence.

This chapter describes OSEMAN in detail. Section 3.1, discusses the object-oriented lexicon and its role in OSEMAN. It starts by finding the correspondence between natural language processing and object-orientation concepts. Section 3.2, gives an overview of OSEMAN. Its internal mechanism is explained by tracing all the phases and transactions carried on from the arrival of a sentence till a semantic representation is produced. The evaluation criterion is also discussed here. Section 3.3, discusses the robustness of OSEMAN and its ability to incorporate various difficulties in semantic analysis. Finally, section 3.4, presents the software development of OSEMAN using use-case driven approach. This approach is based on defining the whole system in terms of actors and use-cases.

3.1 OBJECT-ORIENTED LEXICON

A lexicon is a huge repository, which contains the vocabulary of a language. In any NLP system, lexicon is regarded an essential part. Similarly, OSEMAN relies on an assumed object-oriented lexicon. This section specifies the characteristics of this lexicon.

OBJECT ORIENTATION AND NLP

The main motivation behind designing an object-oriented lexicon is the suitability of object-orientation concepts for modeling the features of natural languages. The vocabulary words of any natural language can be thought of as **objects**. These objects will contain *information* (attributes) and *behavior (methods)* that will reflect the linguistic characteristics of this word. For example, taking the verb '*Fly*' and considering it as an object, there could be attributes like: tense, transitivity, etc. and behaviors such as: Agent, Source, Destination, etc. These attributes and behaviors need not be part of the verb itself, rather it could be inherited from some ancestor verb in the inheritance hierarchy Figure 3.1 illustrates the attributes and behaviors of the object *Fly*.

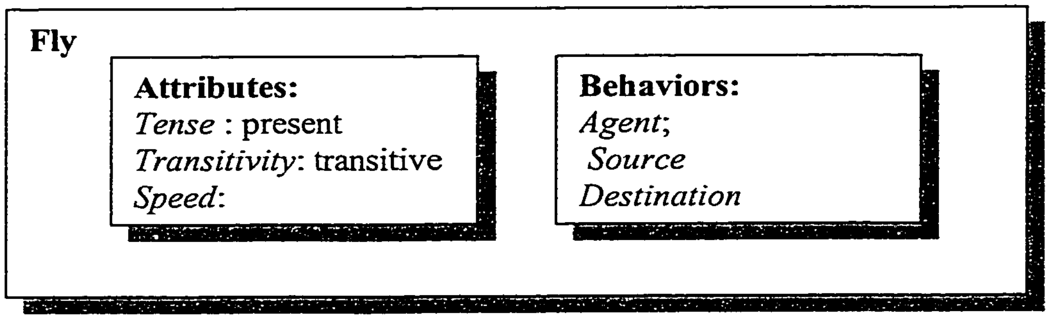


Figure 3.1 Some possible Attributes and Behaviors of the verb *Fly*

A sentence consists of number of objects (words). For example, if the verb 'Fly' appears in a sentence, then the object corresponding to this verb will try to find the agent (who will perform the flight). This is done through invocation of methods that resembles sending messages to other objects in the sentence. Only an object that is semantically suitable to be the agent of the verb Fly will be selected. Similar kind of methods will select suitable objects for source and destination of the verb. Thus, the sentence '*Ahmad flew from Dhahran to Riyadh*' will give '*Ahmad*' as agent, '*Dhahran*' as source and '*Riyadh*' as destination.

Different natural language words (objects) can be grouped into similar classes. For example, *eagle, pigeon* and *canary* can belong to a same class called *Bird*. Thus, *eagle, pigeon* and *canary* will be instances of the class *Bird*. Common behaviors (*fly, hunt, eat, drink*) and attributes (*color of feather, size*) will be defined in the class *Bird*. Each instance will store different values for these attributes. Thus, the instance *Canary* will store 'Yellow' in the color field, whereas; the instance *Eagle* will store 'Brown' as its color

Polymorphism is another object-oriented concept that can be used efficiently in Natural language processing, especially in semantic analysis. By this concept, different subclass instances of the same parent class could respond to a method invocation of the parent class. This method is implemented differently in the two subclasses. As a result, the same method invocation results in different behavior in different subclasses. Thus, through polymorphism, two different verbs belonging to same super class will respond differently for the same method invocation, for example, *agent*. This will add a great flexibility to the system. Using polymorphism many other semantic issues can be addressed.

Inheritance is one of the most useful features of object-orientation that can be exploited by NLP. Semantic hierarchy can be made using inheritance. Taking a noun like: *fly*, the following hierarchy can be made (as taken from WordNet 1.6).

```
fly
     => dipteran
        => insect
           => arthropod
              => invertebrate
                 => animal
                    => organism
                       => entity
```

Figure 3.2 WordNet hierarchy (hypernym) for the noun fly

Similarly taking a verb like: *fly* (as in flying a plane) the following hierarchy can be made (again taken from WordNet 1.6).

```
Fly
    => operate
       => manipulate
          => handle
             => touch
```

Figure 3.3 WordNet hierarchy for the verb Fly

In a similar way all nouns and verbs of a language can be organized semantically. Words will inherent linguistic and semantic features from their parent word(s). Only distinguishing features will be explicitly encoded in the new class. Proceeding in this manner, we end up with an object-oriented lexicon.

BUILDING THE OO LEXICON

Object-orientation features discussed previously could be exploited in designing a lexicon. The lexicon will contain the vocabulary of the language organized in an object-oriented semantic hierarchy. The hierarchy used in WordNet can be adopted. As discussed in the previous chapter, WordNet defines hierarchy for verbs, nouns and adjectives. Each category has more than one root word as the starter of the hierarchy. Verbs have around 14 different starter verbs. Similarly, nouns are partitioned into topical groups semantically. Twenty-five files were created and each partition is started with a unique beginner. The lexicon suggested here will follow the same hierarchy except that another upper layer is introduce to cover all starter verbs, adjectives and nouns. Class Verb and Noun are the root parent of all verbs and nouns. Class Tool includes abstract class definition for adjectives and prepositions. All these root classes inherit from the ultimate root of all words called *Word*. Figure 3.4 (next page) illustrates the class hierarchy structure of the suggested lexicon. The class V*erb* contains certain attributes and methods that are common to all verbs. Tense is an attribute that states the tense of the verb, and associated with it is the `getTense` method. `FindPartner` is another method common to all verbs. This method will be overridden by the children verbs to incorporate appropriate semantic association to find the agent, object and so on. Thus, `FindPartner` invokes another set of methods like `Agent`, `Object`, etc to get the semantic roles performed by the verb filled. The lexicon builds the hierarchy into deeper levels abiding by the same rule, i.e., placing the common attributes and methods in the parent class and letting the children inherit them.

For example, the verb *Move* is inheriting the method `Agent` from its parent: the class *Verb*. The class M*ove* will override the method `Agent` to incorporate source and destination as in the sentence "*Ahmad moved the box from the room to the kitchen*". Descending further in the hierarchy of the verb move, other verbs like *fly, drive, run*, etc. are found.

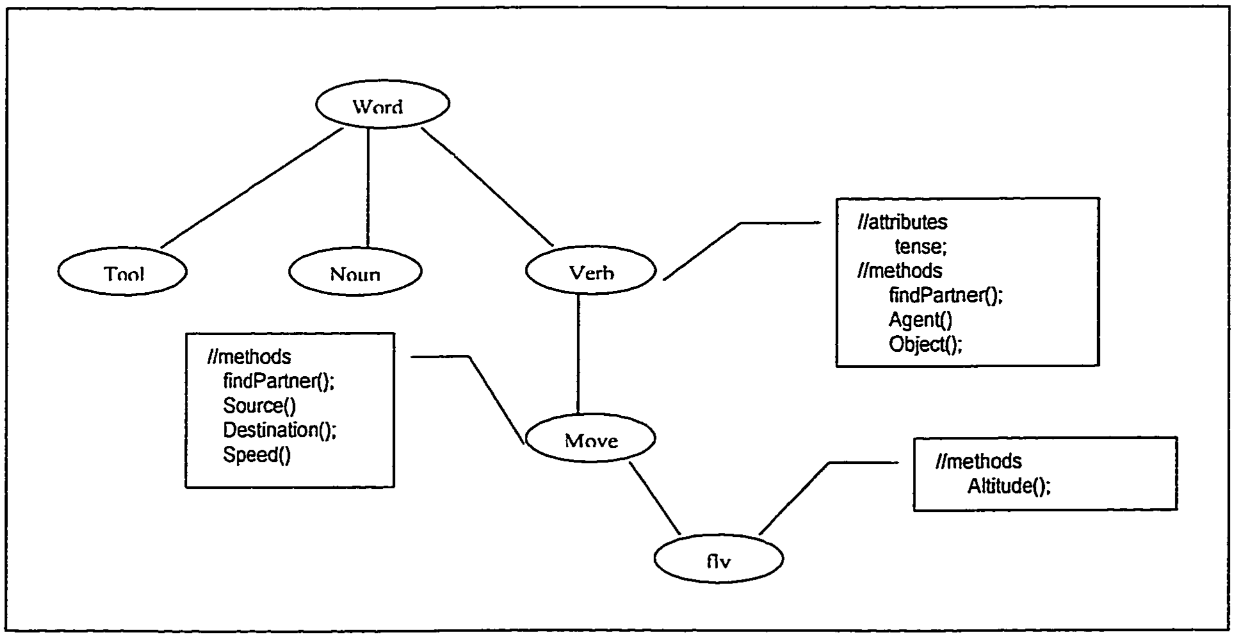


Figure 3.4 Object-Oriented Lexicon Hierarchy

Another method related to verbs is `Associate`. This method is responsible to give the *association values* (V_a) which is described in the next subsection.

In a similar way semantic hierarchy for nouns can be built. In case of nouns, emphasis is made on assigning attributes that will distinguish one class from the others. For example, *Animal* is a noun directly inheriting from the root *Noun*. Attributes are added to *Animal* that will distinguish it from other entities. These characteristics of a particular noun are captured in an attribute of type set_of_strings called `features`. Further going down in the hierarchy of *Animal* two children can be identified: *Human* and *Bird*. The attribute `features` in the class *Human* will contain the set {*talk, smile, read,*....}. Similarly, `features` of *Birds* will be {*fly,...*} . Figure 3.5 depicts the strategy for building the hierarchy for nouns.

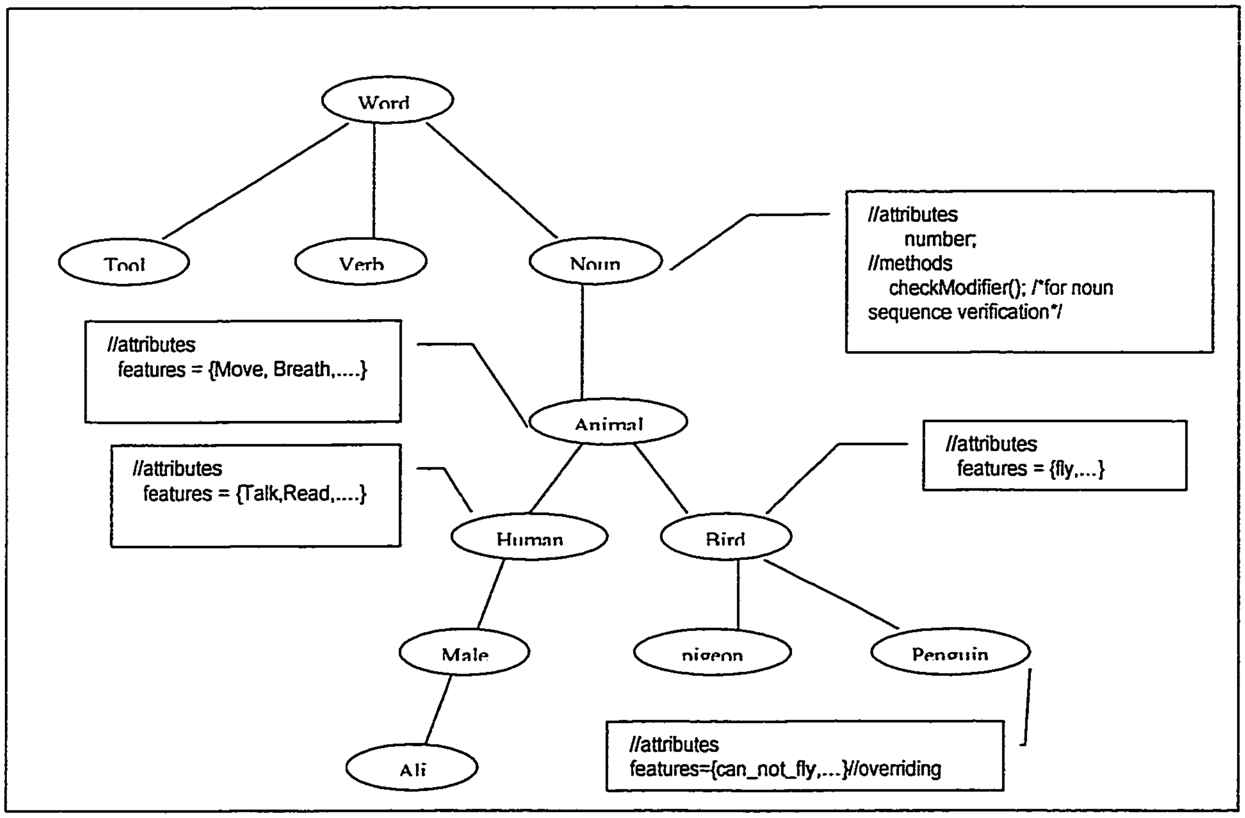


Figure 3.5 Hierarchy of Nouns

In English – similar with any natural language – a word (verb, noun or adjective, etc.) often has more than one meaning or sense. It is important to identify which sense of the word to be considered. The appropriate sense could be deduced from the context of the sentence. Table 3.1 gives different senses for the verb eat as taken from WordNet.

#	Eat	V_r
1.	eat -- (take in solid food; "She was eating a banana"; "What did you eat for dinner last night?")	10
2.	eat -- (eat a meal; take a meal; "We did not eat until 10 P.M. because there were so many phone calls"; "I didn't eat yet, so I gladly accept your invitation")	9.17
3.	feed, eat -- (take in food; used of animals only: "This dog doesn't eat certain kinds of meat"; "What do whales eat?")	8.33
4.	consume, eat up, use up, eat, deplete, exhaust, run through, wipe out -- (use up, as of resources or materials; "this car consumes a lot of gas"; "We exhausted our savings"; "They run through 20 bottles of wine a week")	7.5
5.	eat, eat on -- (worry or cause anxiety in a persistent way: "What's eating you?")	6.67
6.	corrode, eat, rust -- (cause to rust; "The acid corroded the metal")	5.83

Table 3.1 Senses of the verb Eat

It is also observed that these senses are arranged in decreasing order of their popularity. Thus, the first sense is more familiar than the second and so on. Note that sense 5 and 6 are not very closely related to other senses, hence they were at the bottom of the list. To quantify the familiarity of a sense, the following measure gives the rank value (V_r) for a word.

$$\textit{Rank value } (V_r) = \textit{max} - \textit{min} \times \frac{n - 1}{N}$$

Where *n* is the WordNet rank of the sense and N is the total number of senses for this word. *Max* and *min* are two variables that can be controlled to narrow or wide the range of rank values. In all calculations made here, *max* and *min* are given values 10 and 5 respectively. Table 3.2 shows the senses of the verb *time* and their rank values.

#	verb time	V_r
1	clock, time -- (measure the time or duration of an event or action or the person who performs an action in a certain period of time; "he clocked the runners"	10
2	time -- (assign a time for an activity or event	8.75
3	time -- (set the speed, duration, or execution of	7.5
4	time -- (regulate or set the time of, as of a clock or watch	6.25

Table 3.2 Senses of the verb *time* and their rank values (V_r)

The senses are organized in the lexicon according to their hierarchy. Thus, different senses may have different semantic hierarchy. Figure 3.6 gives the hierarchy of the verb *time*.

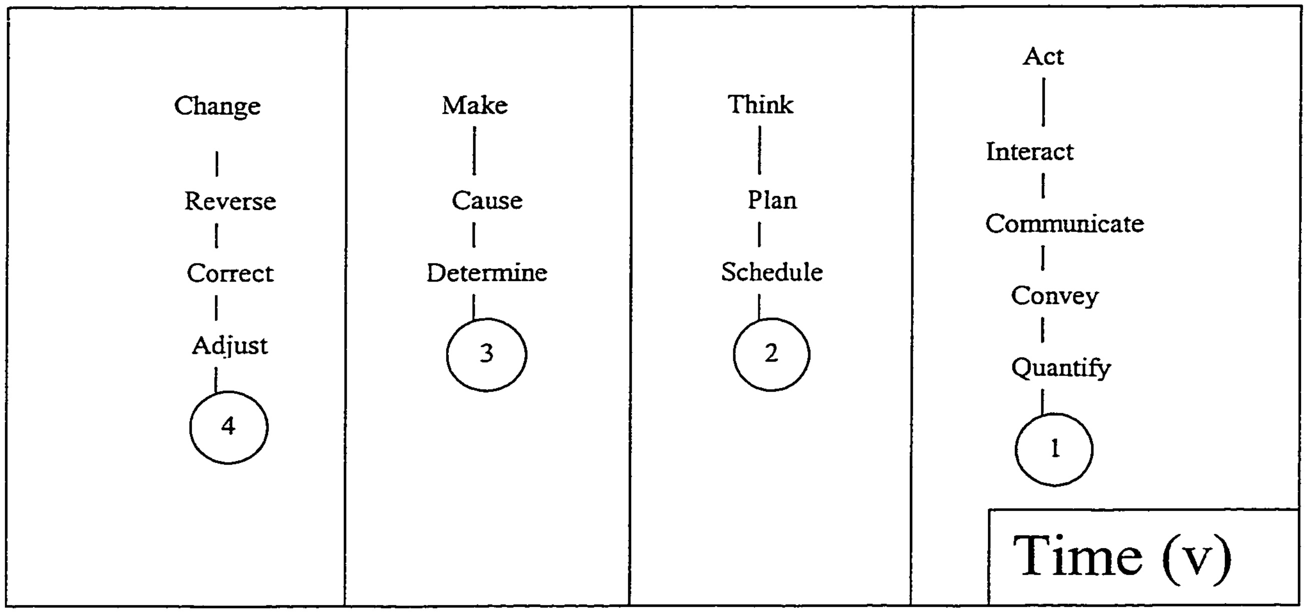


Figure 3.6 Hierarchy of the senses of the verb *time*

3.2 OVERVIEW OF THE MODEL

The model described here (OSEMAN) is based on the object-oriented lexicon described in the previous section. The responsibility of finding an appropriate semantic representation is distributed to objects, which represent words of the system. In the following subsections the phases of the model are discussed.

PHASES OF OSEMAN

Here the behavior of the system is traced from the moment a sentence is arrived till an internal representation that reflects the meaning is produced.

Phase 0

This phase is not part of OSEMAN. This phase is related to morphological analysis, where the surface form of the sentence is converted into a group of stems. It is known that a stem can belong to more than one syntactic category. For example, the stem: *Fly*, could be a verb or a noun. In this early stage it is not possible to consider only one category and neglect the other, hence all syntactic categories of a stem is passed to the next phase. In this phase we will not distinguish between them and hence we cannot rule out any of the possible categories of the stems.

Phase 1

Verbs play a central role in any natural language sentence. Thus a sentence can be described by the verb and its semantic partners. These partners fill different roles assigned by the verb. The sentence: *the boy is eating a banana*, describes an action denoted by the verb *eat*. The complete meaning of the sentence can be obtained after finding semantic partners of *eat*. *The boy* fits well as the **agent** of the verb *eat* and *a banana* suits to be the **object** of the verb *eat*. Thus, the verb, its agent and the object together give an appropriate meaning for the sentence. Given a sentence, it is important to identify which sense of the verb fits well with other partners. For the sentence just mentioned (*the boy is eating banana*) it is clear that the agent *boy* and the object *banana* can only match properly with sense number 1 to give a semantically consisting meaning.

OSEMAN takes the verb in the sentence and instantiates all senses of this verb. Then, each sense will instantiate all senses of a candidate partner and finds the *association value* (V_a) of this partnership. As mentioned in the previous section, the association value is found by the method `Associate` present with every verb. This value is a gray scale value that is in the range of [0,1]. Association value of 1 denotes that the two words are semantically appropriate and Association value of 0 indicates that the two words are semantically inconsistent. Grayscale values in between the range indicates the strong ness of the association. Following section describes how this value is found.. This association is next weighed. This weight must reflect the rank value of the two objects. Thus, the weight is more if the two words associated are in the top position among senses than if they were in the bottom of the list of senses. Hence, the *combination value* (V_c) can be calculated by multiplying the rank value (V_r) of the two words and the binary association value (V_a). Next, the overall weight (W_o) of the sentence for this verb sense is calculated by taking the average value of the individual combination values. The process continues for all senses of the verb. Thus, it is ended with a list of possible combinations along with a weight (W_o) for each combination.

Phase 2

The rest of the process deals with choosing the appropriate combination among the weighted combinations. The straightforward way is to rank these combinations in ascending order and choose the highest weight to represent the meaning of the sentence. To regulate the process a threshold value (V_t) could be used to reject any combination that has less weight.

Phase 3

Finally, the desired combinations are reported to the user in a representation that expresses the semantics of the sentence. For a sentence like, *Ahmed flew from Dhahran to Jeddah*, the internal representation is given in figure 3.7.

```
Action: fly
Tense: past
Agent: Ahmad
Source: Dhahran
Destination: Jeddah
Evaluation weight: 95%
```

Figure 3.7 Internal representation

The steps in the process are summarized in figure 3.8.

```
Input:   stems of the sentence; surface form of the sentence
Output: (an) internal representation(s) that reflect(s) the meaning of the   sentence
steps:

1. For the main verb stem:
        a) Instantiate senses of this verb ($V_r$ will be assigned)
        b) For each sense of this verb:
                I. Instantiate the senses of possible partners
                II. For each partner senses find association value ($V_a$)
                III. For each partner sense find combination value ($V_c$)
                IV. Calculate the average weight of this combination ($W_o$)
2. Reject combinations less than the threshold ($V_t$)
3. Produce the internal representation for accepted combinations
4. Report the internal representation to the end users
```

Figure 3.8 Main Steps of OSEMAN

Figure 3.9 (next page) gives an overview of OSEMAN.

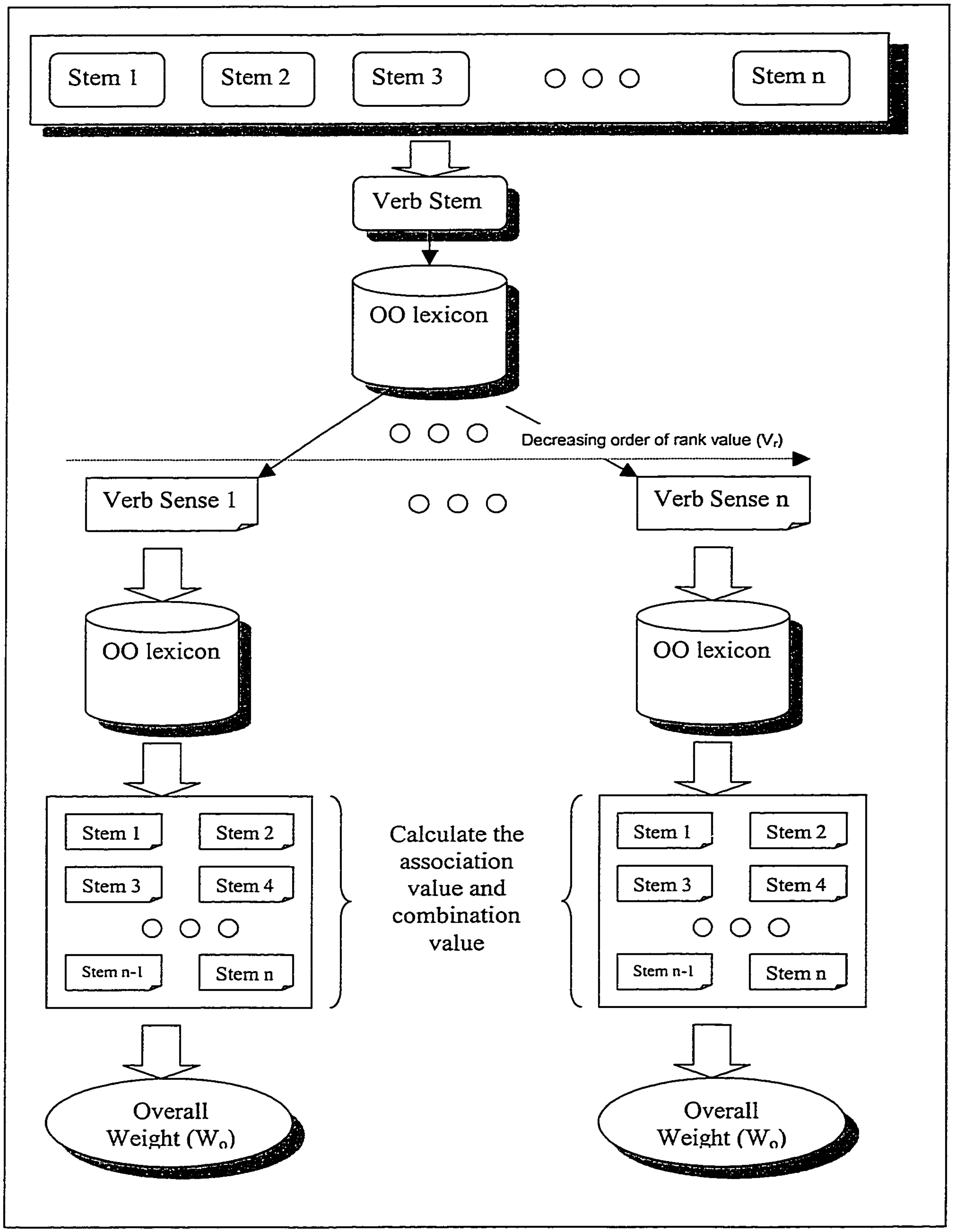


Figure 3.9 Overview of OSEMAN

CALCULATING ASSOCIATION VALUES (V_a)

Association value (V_a) between a verb and a noun is calculated using association function (f_a). This function is a method present in the verb, which was indicated earlier as the method `Associate`. The function takes the attributes of the verb (a_v), and the attributes of the noun (a_n) as input. These attributes are matched, and accordingly a grayscale value in the range [0,1] is returned which reflects the semantic associatively between the two words.

To return a grayscale value, (f_a) checks the correspondence between the types of attributes of the verb with the attributes of the noun. If a direct match is found, then a value of 1 is returned, which indicates that these two words are highly associative. If there is no matching between the verb attribute type and noun attributes, then, the parents of the verb types are checked against the parent of the noun and the process continues until a match is found. The distance is calculated and compared to the total distance from each word to the starter word in the hierarchy. The association value is given accordingly. If the matching found was close to the attribute type, then the grayscale will be closer to 1. On the other hand, if the matching was near the root word, then the returned value will be close to 0.

Example:

Consider association between the second sense of the verb *eat* and the second sense of the noun *banana*. Figure 3.10 (next page) shows the WordNet description of the two words. Figure 3.11 shows a possible object description of these two senses. It is clear that eat expects a valid meal as its object. Banana is not a valid meal. In this case, both the ancestors of meal and banana are checked. The WordNet hierarchy of the word *meal* is given in figure 3.12. Figure 3.13 shows that both words match at the common parent of *food*.

Sense 2
eat -- (eat a meal; take a meal; "We did not eat until 10 P.M. because there were so many phone calls"; "I didn't eat yet, fo I gladly accept your invitation")
=> consume, ingest, take in, take, have -- (serve oneself to, or consume regularly; "Have another bowl of chicken soup!" "I don't

Sense 2
banana -- (elongated crescent-shaped yellow fruit with soft sweet flesh)
=> edible fruit -- (edible reproductive body of a seed plant especially one having sweet flesh)
=> produce, green goods, green groceries, garden truck -- (fresh fruits and vegetable grown for the market)
=> foodstuff, food product -- (a substance that can be used or prepared for use as food)
=> food, nutrient -- (any substance that can be metabolized by an organism to give energy and build tissue)
=> substance, matter -- (that which has mass and occupies space; "an atom is the smallest indivisible unit of matter")
=> object, physical object -- (a physical (tangible and visible) entity; "it was full of rackets, balls and other objects")
=> entity, something -- (anything having existence (living or nonliving))

Figure 3.10 WordNet definition of *eat* and *banana*

Eat (2)
Inherited Attributes:
Agent: causal agent
Object: any_food
Time: any_valid_time
Duration: any_valid_period

Local Attributes:
Agent: human_being;
Object: valid_meal;
Tool_for_Eating: Eating_tool;
Amount_of_Eating: Weight;
Time: valid_meal_time;

Banana (2)
Inherited Attributes:
{has_mass,
used_as_food,
fresh_fruit,
eatable,
sweet}
Local Attributes:
{elongated,
cresent_shaped,
yellow,
soft}

Figure 3.11 Possible Class definition of Eat and Banana

meal, repast -- (the food served and eaten at one time)
=> nutriment, nourishment, sustenance, aliment, alimentation, victuals -- (a source of nourishment)
=> food, nutrient -- (any substance that can be metabolized by an organism to give energy and build tissue)
=> substance, matter -- (that which has mass and occupies space; "an atom is the smallest indivisible unit of matter")
=> object, physical object -- (a physical (tangible and visible) entity; "it was full of rackets, balls and other objects")
=> entity, something -- (anything having existence (living or nonliving))

Figure 3.12 WordNet definition of *meal*

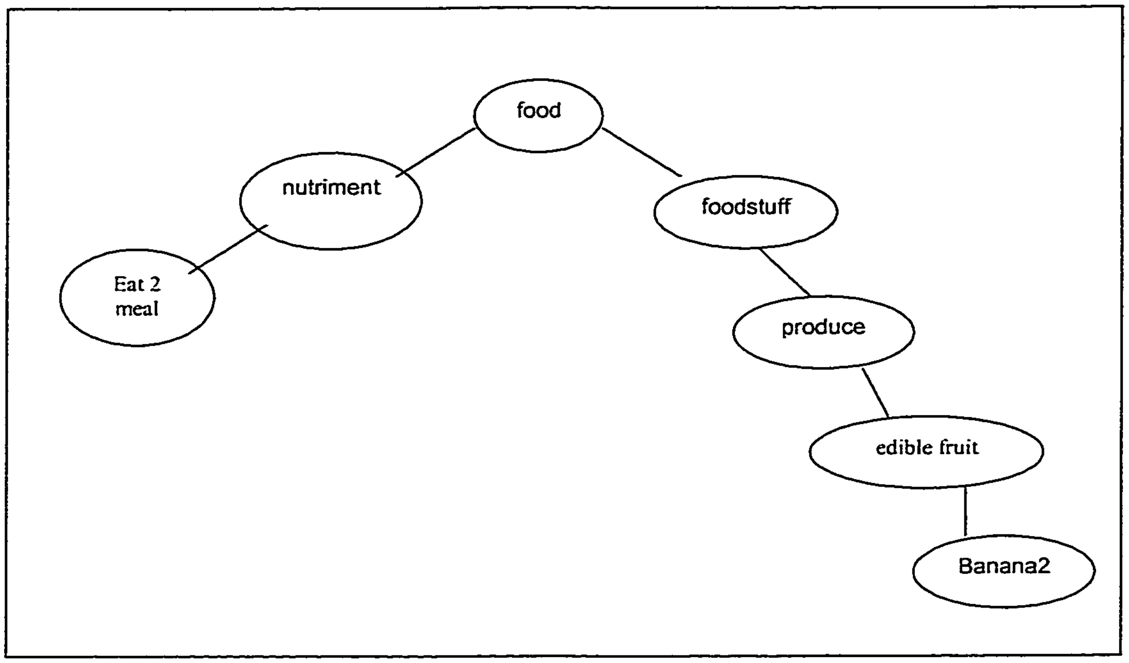


Figure 3.13 Association hierarchy between eat and banana

To calculate Va, it is evident that the total length of the hierarchy of the two words (where both share the common starter of entity) is 12. The length for the association where the common parent is food is 6. Therefore, the association value will be 6/12 = 0.5.

3.3 HANDLING DIFFERENT ISSUES IN SEMANTIC ANALYSIS

In chapter 2, some challenges and difficulties of semantic analyzer were discussed. This section shows how OSEMAN overcomes these difficulties. OSEMAN is based on the idea that sentence words are objects possessing some attributes and behaviors. These attributes and behaviors are chosen in a way to handle various semantic as well as linguistic issues. OSEMAN is flexible enough to accommodate new problems and issues. In such case, features of Object-Orientation can be exploited such as addition of new methods and attributes.

Finding Agents and Objects & Other semantic roles

Agents and objects are two features that almost every verb has. Agent describes who performs the action denoted by the verb. In passive sentences the agent could be hidden. An agent must be semantically in consistent with the verb. In OSEMAN, each verb has a method named `Agent` that gives the agent of this verb. Considering a candidate for agent, the method checks for semantic and syntactic consistency of this candidate before assigning 1 as an *association value.*

Object defines on whom the verb performs the action. Objects are properties of every transitive verbs. In OSEMAN, a method called `Object` is encoded for every verb. Only transitive verbs will have a valid implementation for that. The implementation varies from one verb to other. The feature of overriding in Object Orientation is used for that. This method checks for semantic as well as syntactic consistency of a candidate object before accepting it and assigning 1 as an association value. Some transitive verbs expect two objects or even more. This method handles this property as well.

Many nouns in the sentence other than agent or object give different features of the verb. Often these words are coming after prepositions. Taking a verb like *move* and all its descendents, properties like source, destination, direction of movement, speed, tool for movement, etc all could be present in the sentence. These attributes are often preceded by prepositions like: *from* before a source and *to* before a destination. The lexicon contains abstract classes for all prepositions. These classes have methods that find appropriate nouns that will fulfill the semantic roles assigned by these prepositions. Thus, the preposition *from* picks the appropriate word to be the source and the preposition *to* also picks the appropriate noun that represents the role of destination.

Handling Adjectives and Adverbs

Adjectives describe nouns. They are encoded in the lexicon as Abstract classes. These classes are not instantiated into objects. Rather, they contain methods, which associate some attributes to the noun they describe. Similarly, adverbs describe verbs. They correspond to some abstract classes in the lexicon and contain methods that describes some attributes in the verb they describe.

Handling Verbal Polysemy and Word Sense Disambiguation

In case of verbal polysemy, a single verb will have more than one meaning depending on the context. These are different senses of the same verb. The lexicon maintains different object for each sense. These senses may belong to different hierarchy, and thus attributes and methods could be different for different senses. OSEMAN instantiates each sense and validates it as explained in the phases of OSEMAN. For example, the verb *take* could mean *buy*, *spend* or *obtain* and it is not possible to disambiguate without looking to the object of the verb. Thus, "take *stake*" means "buy stake", and "take *time*" means "spend time" and "take a *product*" means "obtain a product".

Handling Noun Sequence Problem

In a running text, when two nouns in a sequence are encountered, then there is a semantic relation between these two nouns. The meaning becomes obvious after discovering this relation. For example, in the noun sequence "*field mouse*" the modifier *field* describes the location of the head: *mouse*. Whereas, in the NS "*night attack*" the modifier describes the *time* of the head. To handle noun sequences, OSEMAN could be extended. When the sentence contains two or more nouns in sequence, OSEMAN triggers noun sequence handler module. The handler considers the last noun and the head noun. The head noun will associate with the modifier and an association value is given that reflects the semantic consistency of the noun sequence. In this way appropriate weight can be assigned for noun sequences.

3.4 SOFTWARE DEVELOPMENT OF OSEMAN

System development is a complex task. It involves tackling many requirements needed by the system, and thus, it is difficult to handle these requirements simultaneously. What is needed is to handle complexity in an organized way. This is done by working with different models, each focusing on a certain aspect of the system. By reducing the complexity gradually in a specific order in successive models, it is possible to manage the system complexity.

Software development process requires five stages [7][8]. **Requirements model** aims to capture the functionality of the system. **Analysis model** aims to give the system a robust and changeable object structure. **Design model** aims to adopt and refine the object structure to the current implementation environment. **Implementation model** aims to implement the system. The **test model** aims to verify the system.

Use-case model goes through the above-mentioned stages of software development. It starts with requirements specification through using **actors** and **use-cases**. These concepts are simply an aid to defining what exists outside the system (actors) and what should be performed by the system (use-cases). Actors represent everything that needs to exchange information with the system. When a user (an instance of an actor) uses the system, he or she will perform a behaviorally related sequence of transactions in a dialogue with the system. Such a special sequence is called a use-case. Each use-case is a specific way of using the system and every execution of the use-case may be viewed as an instance of the use-case. When a user inputs a stimulus, the use-case instance executes and starts a transaction belonging to the use-case. Thus, in object-oriented a use-case can be thought of as a class description. And the set of all use-case descriptions specifies the complete functionality of the system. A use-case diagram helps in visualizing the behavior of the whole system in terms of actors and use-cases.

The use-case model advocates finding classes for a system under development by looking for boundary, control and entity classes. Entity classes model information and responsibilities of a use-case. Boundary classes are used to model the system interfaces. Control classes model sequencing behavior specific to one or more use-cases. They represent the dynamics of the use-case and they are responsible for the flow of events in the use-case. All these classes can be depicted using **class diagrams**.

In the design stage interactions among objects need to be identified. The use-case diagram presents an outside view of the system. The functionality of the use-case is captured in the flow of events. Scenarios are used to describe how use-cases are realized as interactions among societies of objects. A scenario is an instance of a use-case. Scenarios are developed to help identify the objects, the classes, and the object interactions needed to carry out a piece of the functionality specified by the use-case. They show how the responsibilities specified in the use-cases are distributed among the objects and classes in the system. These object interactions arranged in time sequence is depicted through **sequence diagrams**.

USE-CASE MODEL FOR OOSEMAN

The first step is to discover actors in the system. Actors are someone or something external to the system that must interact with the system under development. They provide input or receive output from the system. Semantic analyzer receives stems from morphological analyzer, hence it will be an actor. Users will be notified about errors and the evaluation of the internal representations. Discourse analyzer will receive the internal representation of the input sentence. Thus user and discourse analyzer will be two appropriate actors. Lexicon will be a part of the system and hence it will not be an actor of our system. Table 3.3 (next page) summarizes the actors in OSEMAN.

Actor	description
Morphological analyzer	Produce stems
User	Receive error messages
Discourse analyzer	Receive the internal representation

Table 3.3 Actors in OOSEMAN

Next, use-cases are identified. A use-case will represent a complete course of action from beginning to end. The main use-case in the system is called: *build internal representation*. This use-case is responsible for all actions starting from instantiating stems till evaluating and ranking the internal representations. There will be an extended use-case for generating error messages to the user. Figure 3.14 gives an overall picture of the system through use-case diagram.

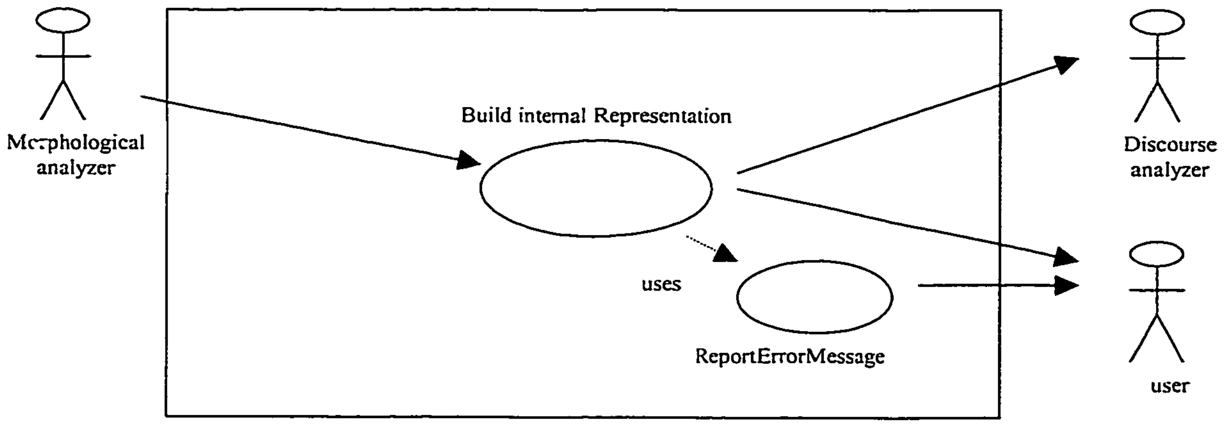


Figure 3.14: Use-case diagram of OSEMAN

By defining actors and use-cases, the requirements of the semantic analyzer are well specified. Next these requirements are investigated more by discovering classes associated with each use-case. As mentioned previously, there are three types of classes: entity, control and boundary class. Here the classes in the main use-case: *build internal representation* are discussed. The stems produced by the actor: morphological analyzer is entertained by a boundary class called: MAInterface. These stems are passed to a control class called: SemanticAnalyzerManager which instantiates the verb from the lexicon and is represented by an entity class called: Verb. Different senses of the Verb are associated with different senses of other stems. this is handled by another entity class called: combination. Each possible combination carries some weight representing its likelihood to reflect the sentence meaning. A control class called: evaluationManager ranks these evaluations, and another boundary class called UserInterface presents the result to the user. Following table summarizes the classes in the main use-case.

Classname	Type	Description
MAInterface	Boundary	Receive stems
UserInterface	Boundary	Present the internal representation and the evaluation to the user
SemanticAnalyzerManager	Control	Instantiate stems as objects from the lexicon.
Error message	Entity	Produce error code
Verb	Entity	Contains the verb and handles association
Combination	Entity	Produce internal representation after relating the stems.
EvaluationManager	Control	Rank the internal representations according to their most likely interpretations.

Table 3.4 Classes in the use-case: *build internal representation*

Figure 3.14 (next page) shows the sequence diagram for the main use-case.

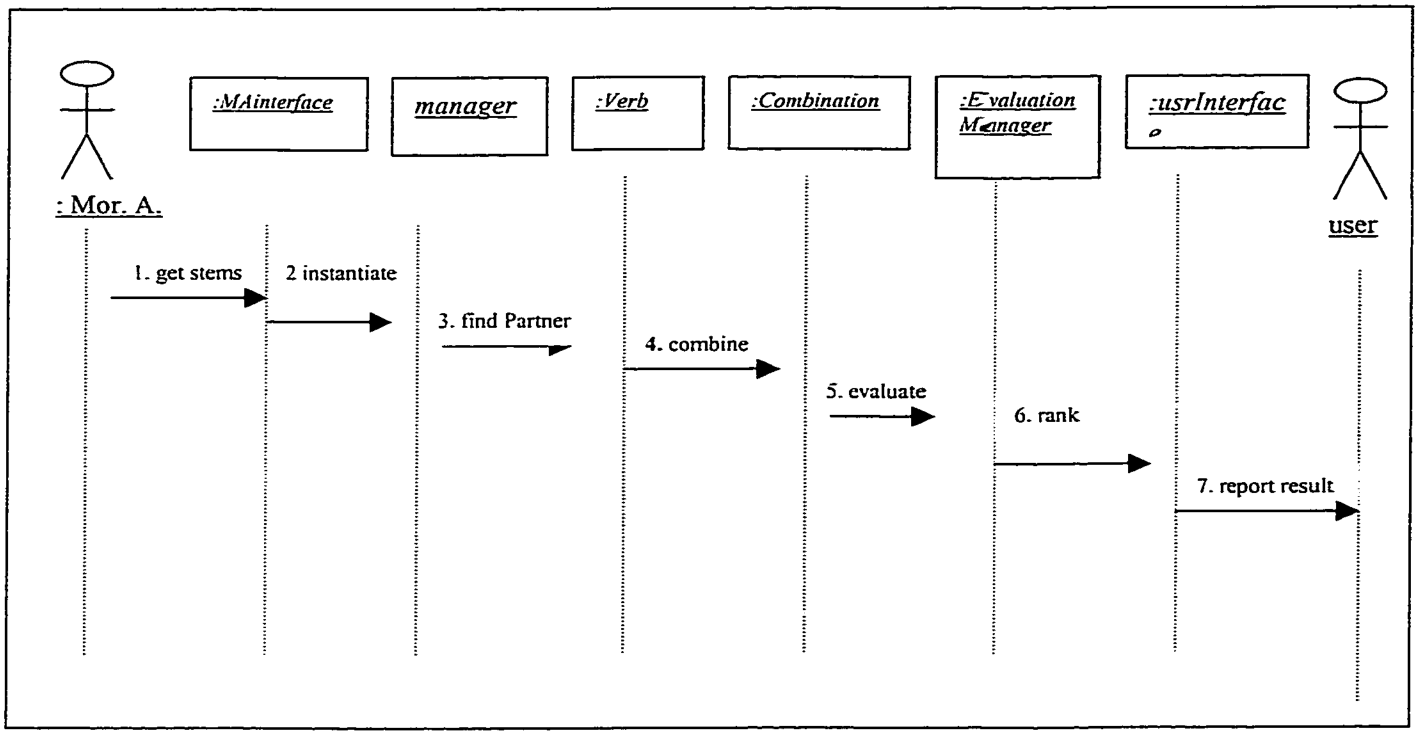


Figure 3.15: Sequence Diagram for 'build internal representation' use-case

CHAPTER 4

CASE STUDIES

This chapter studies few sentences and verifies the feasibility of the model against these sentences. Four sentences are chosen in a way that demonstrates various linguistic and semantic issues.

4.1 SENTENCE 1: THE BOY IS EATING A BANANA

This sentence is fairly simple and straightforward. It resembles the structure of most of the simple English sentences, where it consists of a subject, verb and object. Moreover, this sentence is unambiguous, i.e., it conveys just one unique meaning. This is true because each of the stems in this sentence belong to only one syntactic category. However, each stem gives different –yet related- senses. Table 4.1 gives WordNet overview of these stems along with *rank value* (discussed in Chapter 3) of each sense.

#	Overview of the Noun Boy	V_r
1.	male child, boy, child -- (a youthful male person; "the baby was a boy"; "she made the boy brush his teeth every night"; "most soldiers are only boys in uniform"	10
2.	boy -- (a friendly informal reference to a grown man; "he likes to play golf with the boys"	8.75
3.	son, boy -- (a male human offspring; "their son became a famous judge"; "his boy is taller than he is"	7.5
4.	boy -- (offensive term for Black man; "get out of my way boy"	6.25
	Overview of the Verb Eat	
7.	eat -- (take in solid food; "She was eating a banana"; "What did you eat for dinner last night?"	10
8.	eat -- (eat a meal; take a meal; "We did not eat until 10 P.M. because there were so many phone calls"; "I didn't eat yet, so I gladly accept your invitation"	9.17
9.	feed, eat -- (take in food; used of animals only: "This dog doesn't eat certain kinds of meat"; "What do whales eat?"	8.33
10.	consume, eat up, use up, eat, deplete, exhaust, run through, wipe out -- (use up, as of resources or materials; "this car consumes a lot of gas"; "We exhausted our savings"; "They run through 20 bottles of wine a week"	7.5
11.	eat, eat on -- (worry or cause anxiety in a persistent way: "What's eating you?"	6.67
12.	corrode, eat, rust -- (cause to rust; "The acid corroded the metal"	5.83
	Overview of the Noun Banana	
1.	banana, banana tree -- (any of several tropical and subtropical treelike herbs of the genus Musa having a terminal crown of large entire leaves and usually bearing hanging clusters of elongated fruits	10
2.	banana -- (elongated crescent-shaped yellow fruit with soft sweet flesh	7.5

Table 4.1 WordNet Overview of the stems: Boy, Eat and Banana

The first step in the process of analyzing this sentence is to pass it through a morphological analyzer, which will return the stems of the sentence. Thus the **stems** are boy, eat, banana. Next, the verbs are identified. In this sentence the only verb is *eat*. Next, for each sense of the verb, it is first initialized from the object-oriented lexicon. Then, it makes association with other stems present in the sentence. This association is two-way association. This means, the association between the verb sense and the other stem sense is either accepted or rejected. This decision is done according to semantic consistency. For example, the first sense of *eat* (take in solid food) when associated with the first sense of *banana* (a banana tree), gives inconsistent meaning, thus it should be rejected. The class definition of the object *eat* requires that any object of this verb should be something eatable. In the same way all senses of the verb is examined with all senses of other possible partners. Whenever certain association gives a value of zero it is rejected. When the association value for a pair is 1, the combination value for this pair will be the multiplication of the rank values of the two stem senses. Table 4.2 shows the *association values* of all possible combinations of the stem senses. Table 4.3 shows the *combination values* of these combinations.

			boy				banana	
			1	2	3	4	1	2
			10	8.75	7.5	6.25	10	7.5
eat	1	10	1	1	1	1	0	1
	2	9.17	1	1	1	1	0	0
	3	8.33	0	0	0	0	0	1
	4	7.5	1	1	1	1	1	1
	5	6.67	1	1	1	1	0	0
	6	5.83	0	0	0	0	0	0

Table 4.2 Association values (V_a) for the combinations in Sentence 1

			boy				banana	
			1	2	3	4	1	2
			10	8.75	7.5	6.25	10	7.5
eat	1	10	100	87.5	75	62.5	0	75
	2	9.17	91.7	80.2	68.8	57.3	0	0
	3	8.33	0	0	0	0	0	62.5
	4	7.5	75	65.6	56.3	46.9	75	56.3
	5	6.67	66.7	58.4	50	41.7	0	0
	6	5.83	0	0	0	0	0	0

Table 4.3 Combination value (V_c) for the combinations in Sentence 1

After determining the combination value of each combination, the overall weight (W_o) of the sentence with this combination is calculated. This is nothing but the average of the total number of combinations of the pairs. Table 4.4 shows the total combinations with their perspective weights. Note we can use appropriate value for threshold to control the degree of the reported interpretations. Also note that the greatest weight points to the most likely interpretation.

Eat		Boy			Banana			Weight	Comment
1	10	1	10	100	2	7.5	75	87.5	The boy is eating banana fruit
		2	8.75	87	2	7.5	75	81	The young man is eating banana fruit
		3	7.5	75	2	7.5	75	75	The son is eating banana fruit
		4	6.25	62.5	2	7.5	75	68.8	The black man is eating banana fruit
4	7.5	1	10	75	1	10	75	75	The boy is consuming banana tree
					2	7.5	56.3	65.7	The boy is consuming banana fruit
		2	8.75	65.6	1	10	75	70.3	The young is consuming banana tree
					2	7.5	56.3	60.9	The young is consuming banana fruit
		3	7.5	56.3	1	10	75	65.7	The son is consuming banana tree
					2	7.5	56.3	56.3	The son is consuming banana fruit
		4	6.25	46.9	1	10	75	60	The black man is consuming banana
					2	7.5	56.3	51.6	The black man is consuming banana

Table 4.4 Combinations and weights for sentence 1

Discussion

Different senses of the verb *eat* are instantiated from the object-oriented lexicon. Each sense is an object that contains attributes and methods. Among these methods, there are methods that searches for appropriate agent and object (for transitive verbs) for the verb. The search will consider any word that appears in the sentence regardless of its spatial position in the sentence. Thus, the a sense of the verb eat will consider both boy and banana as possible candidate to fill the role of an agent, then it will reject banana (by assigning a value 0 for association value). Similarly, both boy and banana will be considered as an object of eat and later only the latter is accepted.

The user can set certain constrains to control the quality and type of semantic checking. If in ordinary case a boy is not an eatable thing, in the domain of fair tales, for example, this could be an acceptable association.

Threshold is another variable that can be controlled by the user to narrow or wide the range of acceptable interpretations. It is not desired to have just one interpretation at this stage. On the other hand, too many interpretations will also slow the process. In this example, setting the threshold at 75, for example, will result in a reasonable number of interpretations. In subsequent stages the possible interpretations will further be narrowed.

4.2 SENTENCE 2: TIME FLIES LIKE AN ARROW

This sentence was discussed in chapter 2. It was shown that this sentence is ambiguous. Three different interpretations of this sentence was mentioned and is given once more:

1. Time passes along in the same manner as an arrow gliding through space.
2. I order you to take timing measurements on flies, in the same manner as you would time an arrow.
3. Fruit flies like to feast on a banana; in contrast, the species of flies known as "time flies" like an arrow.

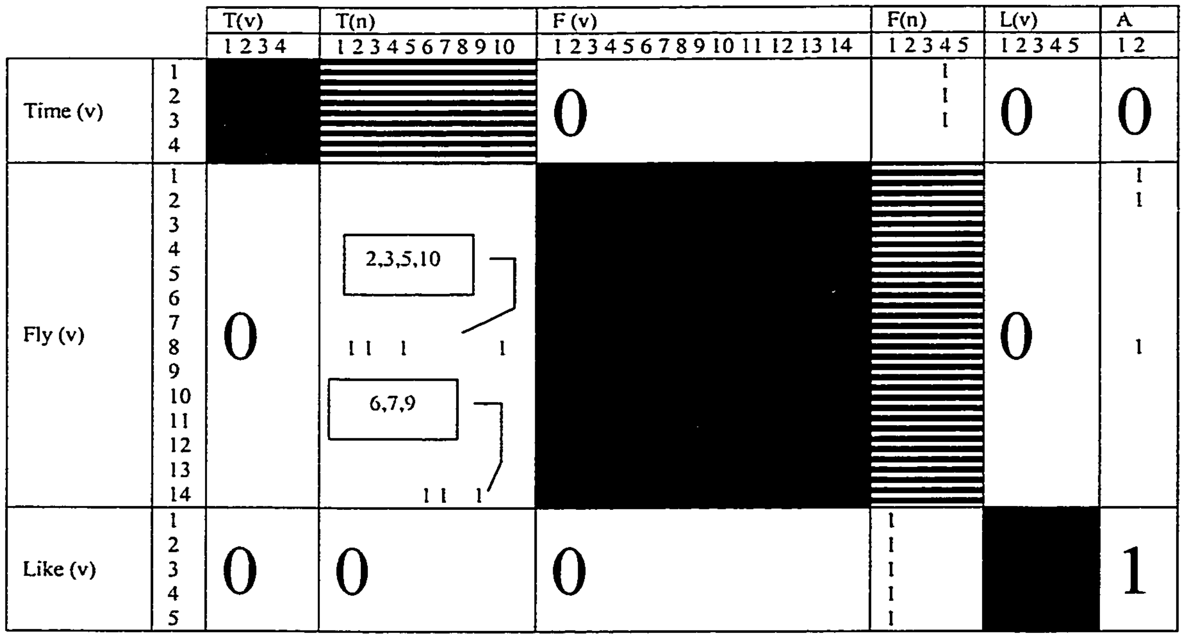
The stems are: *time, fly, like* and *arrow*. Each of the first three stems belongs to two syntactic categories.

#	Overview of noun time	Vr
1	time, clip -- (an instance or single occasion for some event; "This time he succeeded"; "He called four times"; "he could do ten at a clip"	10
2	time -- (an indefinite period (usually marked by specific attributes or activities) ; "he waited a long time"; "the time of year for planting"; "he was a great actor is his time"	9.5
3	time -- (a period of time considered as a resource under your control and sufficient to accomplish something; "take time to smell the roses"; "I didn't have time to finish"; "it took more than half my time"	9.0
4	time -- (a suitable moment; "it is time to go"	8.5
5	time -- (the continuum of experience in which events pass from the future through the present to the past	8.0
6	clock time, time -- (the time as given by a clock; "do you know what time it is?"; "the time is 10 o'clock"	7.5
7	fourth dimension, time -- (the fourth coordinate that is required (along with three spatial dimensions) to specify a physical event	7.0
8	time -- (a person's experience on a particular occasion; "he had a time holding back the tears" or "they had a good time together"	6.5
9	meter, time -- (rhythm as given by division into parts of equal time	6.0
10	prison term, sentence, time -- (the period of time a prisoner is imprisoned; "he served a prison term of 15 months"; "his sentence was 5 to 10 years"; "he is doing time in the county jail"	5.5
	Overview of verb time	
1	clock, time -- (measure the time or duration of an event or action or the person who performs an action in a certain period of time; "he clocked the runners"	10
2	time -- (assign a time for an activity or event	8.75
3	time -- (set the speed, duration, or execution of	7.5
4	time -- (regulate or set the time of, as of a clock or watch	6.25
	Overview of noun fly	
1	fly -- (two-winged insects characterized by active flight	10
2	tent-fly, fly sheet, fly, tent flap -- (a piece of canvas that can be drawn back to provide entrance to a tent	9
3	fly, fly front -- (a garment closure (zipper or buttons concealed by a fold of cloth	8
4	fly, fly ball -- (the act of hitting a baseball so that it flies high in the air	7
5	fly -- ((angling) fisherman's lure; a fishhook decorated to look like an insect	6
	Overview of verb fly	
1	fly, wing -- (travel through the air; be airborne; "Man cannot fly"	١٠
2	fly -- (move quickly or suddenly; "He flew about the place"	9.64
3	fly, aviate, pilot -- (fly a plane	9.28
4	fly -- (transport by aeroplane; "We fly flowers from the Caribbean to North America"	8.92
5	fly -- (cause to fly or float: "fly a kite"	8.57
6	fly -- (be dissipated; "Rumors and accusations are flying"	8.21
7	fly -- (change quickly from one emotional state to another: "fly into a rage"	7.85
8	fly, fell, vanish -- (pass away rapidly; "Time flies like an arrow"; "Time fleeing beneath him"	7.50
9	fly -- (travel in an airplane; "she is flying to Cincinnati tonight"; "Are we driving or flying?"	7.14
10	fly -- (display in the air or cause to float: "fly a kite"; "All nations fly their flags in front of the U.N."	6.78
11	flee, fly, take flight -- (to run away: "He threw down his gun and fled."	6.42
12	fly -- (travel over (an area of land or sea in an aircraft; "Lindbergh was the first to fly the Atlantic"	6.07
13	fly -- (hit a fly, in baseball	5.71
14	vanish, fly -- (decrease rapidly, as of money	5.35
	Overview of verb like	
1	wish, care, like -- (prefer or wish to do something; "Do you care to try this dish?" "Would you like to come along to the movies?"	10
2	like -- (find enjoyable or agreeable; "I like jogging"; "She likes to read Russian novels"	9
3	like -- (be fond of; "I like my nephews"	8
4	like -- (feel about or towards; consider, evaluate, or regard; "How did you like the President's	7

	speech last night?"	
5	like – (want to have; "I'd like a beer now!"	6
	Overview of adj like	
1	like, similar – (resembling or similar; having the same or some of the same characteristics; often used in combination; "suits of like design"; "a limited circle of like minds"; "members of the cat family have like dispositions"; "as like as two peas in a pod"; "doglike devotion"; "a dreamlike quality"	10
2	like, equal, equivalent, same – (equal in amount or value; "like amounts"; "equivalent amounts"; "the same amount"; "gave one six blows and the other a like number"; "an equal number"; "the same number"	8.75
3	alike, similar, like – (having the same or similar characteristics; "all politicians are alike"; "they looked utterly alike"; "friends are generaly alike in background and taste"	7.5
4	comparable, corresponding, like – (conforming in every respect; "boxes with corresponding dimensions"; "the like period of the preceding year"	6.25
	Overview of noun arrow	
1	arrow, pointer – (a mark to indicate a direction or relation	10
2	arrow – (a projectile with a straight thin shaft and an arrowhead on one end and stabilizing vanes on the other; intended to be shot from a bow	7.5

Table 4.5 WordNet overview of the stems in sentence 2

Table 4.6 shows the association value (A) between the verbs in this sentence and other possible stems. Table 4.7 gives the different interpretations of this sentence when the threshold is 50.



T(v) 1 2 3 4
T(n) 1 2 3 4 5 6 7 8 9 10
F (v) 1 2 3 4 5 6 7 8 9 10 11 12 13 14
F(n) 1 2 3 4 5
L(v) 1 2 3 4 5
A 1 2
Time (v) 1 2 3 4
Fly (v) 1 2 3 4 5 6 7 8 9 10 11 12 13 14
Like (v) 1 2 3 4 5
0
2,3,5,10
1 1 1 1
6,7,9
1 1 1
1

Table 4.6 Association values for sentence 2

Fly (v)		Time (n)			Arrow (n)			Weight	Comment
8	7.50	2	9.5	71.3	2	7.5	56.3	**63.8**	Time2 passes as arrow passes
		3	9.0	67.5	2	7.5	56.3	61.9	Time3 passes as arrow passes
		5	8.0	60	2	7.5	56.3	58.2	Time5 passes as arrow passes
		10	5.5	41.3	2	7.5	56.3	48.8	Time10 passes as arrow passes

Table 4.7 Possible Interpretations and Weights of sentence 2

Discussion

Among the three different interpretations mentioned, the model reports only one interpretation with different senses that are related. This is the case because the other two interpretations, though possible, semantically inconsistent. Fly (the insect) is not something that can be timed (measured). And the hypothetical fly (time-fly) does not exist and if it does, is not fond of arrows. Thus, all these associations are rejected earlier by the model and their association value are 0 as shown in table 4.6.

The interpretation that considers '*time flies*' as a species of flies, raises the issue of noun sequence which is discussed in chapter 3. This could be handled by encoding *time_flies* as an object in the object-oriented lexicon. Another automatic way to recognize noun sequences is to let nouns associate together semantically to find possible noun sequence relation. This method is discussed in detail in chapter 3.

The word *like* in the sentence plays the role of an adjective. It describes that the concept mentioned previously is similar to the concept mentioned after. Thus, *time* passes in a similar way as *arrow* passes. The lexicon will not consider adjectives as separate objects. They will be abstract classes. These classes contain abstract methods which will find the appropriate semantic roles associated with this adjective. The abstract methods in the class like will find the two parts that are similar to each other.

4.3 SENTENCE 3: THE BOY IS EATING ICE CREAM WITH A SPOON

The WordNet overview of ice cream and spoon is given in table 4.8 below.

#	Overview of noun spoon	V_r
1	spoon -- (a piece of cutlery with a shallow bowl-shaped container and a handle; used to stir or serve or take up food)	10
2	spoon, spoonful -- (as much as a spoon will hold; "he added two spoons of sugar")	8.3
3	spoon -- (formerly a golfing wood with an elevated face)	6.7
	Overview of verb spoon	
1	spoon -- (scoop up or take up with a spoon; "spoon the sauce over the roast")	10
2	smooch, snog, spoon -- (cuddle and kiss)	7.5
	Overview of noun ice-cream	
1	ice cream -- (frozen dessert containing cream and sugar and flavoring)	10

Table 4.8 WordNet overview of the stems in sentence 3

Table 4.9 depicts the association value of the verbs with other nouns. Table 4.10 shows the reported interpretation with their weight.

		Boy	Eat	Ice cream	Spoon(n)	Spoon(v)
		1 2 3 4	1 2 3 4 5 6	1	1 2 3	1
eat	1	1 1 1 1	■	1	1 1 0	0
	2	1 1 1 1		0	0 0 0	
	3	0 0 0 0		0	0 0 0	
	4	1 1 1 1		0	0 0 0	
	5	1 1 1 1		0	0 0 0	
	6	1 1 1 1		0	0 0 0	
spoon	1	1	0	1	1 0 0	■
	2			0	0 0 0	

Tabel 4.9 Association value of the stems in sentence 3

Eat		Boy			Ice cream			spoon			Weight	Comment
1	10	1	10	100	1	10	100	1	10	100	100	The boy is eating ice cream with a spoon
		2	8.75	87.5	1	10	100	1	10	100	95.8	The young man is eating ice cream with a spoon
		3	7.5	75	1	10	100	1	10	100	91.7	The son is eating ice cream with a spoon
		4	6.25	62.5	1	10	100	1	10	100	87.5	The black man is eating ice cream with a spoon

Table 4.10 Possible interpretations of sentence 3 and their weights

Discussion

This sentence demonstrates the issue of *prepositional phrase* (PP) attachment. The PP *with spoon* could be attached with ice-cream or it could be attached with *eat*. The model finds that the association between the verb *eat* and *spoon* is accepted as a tool for eating. This association is made by the abstract methods in the abstract class: W*ith*. Consider the sentence 'the boy is eating ice cream with vanilla'. Here PP with vanilla is attached with ice cream and not with eat. In this case the class *With* accepts ice-cream to be the appropriate partner of vanilla, and rejects the verb eat. Thus, in both cases the sentence is unambiguous and only one interpretation is considered.

4.4 SENTENCE 4: THE BOY SAW THE MAN ON THE MOUNTAIN WITH A TELESCOPE

This sentence is ambiguous. Prepositions: *on* and *with* allowed this sentence to have more than one interpretation. The interpretations are:

1. The boy used telescope to see the man who was on the mountain
2. The boy who was on the hill used telescope to see the man
3. The boy saw the man who was on the hill carrying a telescope
4. The boy saw the man on the mountain, and this mountain was carrying a telescope. (Somehow absurd meaning)

Following are tables showing WordNet overview of some the stems appearing in the sentence.

SEE (verb)	V_r
1. see -- (perceive by sight; "You have to be a good observer to see all the details"; "Can you see the bird in that tree?"	١٠
2. understand, realize, see -- (perceive mentally, as of an idea; "Now I see!"; "I just can't see your point"	9.761
3. witness, find, see -- (perceive with any or all of one's senses; "We found Republicans winning the offices"; "You'll see a lot of cheating in this school"; give rise to or be characterized by; "The 1960 saw the rebellion of the younger generation against established traditions"	9.523
4. visualize, envision, project, fancy, see, figure, picture, image -- (imagine; see in one's mind; "I can't see him on horseback!" "I can see what will happen"	9.285
5. see, consider, reckon, view, regard -- (consider or deem to be; regard; "She views this quite differently from me"; "I consider her to be shallow"	9.047
6. learn, hear, get word, get wind, pick up, find out, get a line, discover, see -- (get to know or become aware of; "I learned that she has two grown-up children"; "I see that you have been promoted"	8.809
7. watch, view, see, catch, take in -- (see or watch; "view a show on television"; "This program will be seen all over the world"; "view an exhibition"; "Catch a show on Brodaway"	8.571
8. meet, ran into, forgather, foregather, encounter, run across, come across, see -- (meet; "I'll probably see you at the meeting"; "How nice to see you again!"	8.333
9. determine, check, find out, see, ascertain, watch, learn -- (find out or learn with certainty; "I want to see whether she speaks French"; "See whether it works"; "Watch how he will react"	8.095
10. see, check, insure, see to it, ensure, control, ascertain, assure -- (be careful or certain to do something; make certain of something; "He verified that the valves were closed"; "See that the curtains are closed"; "control the quality of the product"	7.857
11. see -- (go to see for professional or business reasons; "You should see a lawyer"; "We had to see a	7.619

psychiatrist"	
12. see -- (go to see for a social visit; "I went to see my friend Mary the other day"	7.380
13. visit, see -- (visit a place, as for entertainment; "We went to see the Eiffel Tower in the morning"	7.142
14. attend, take care, look, see -- (take charge of; "Could you see about lunch?"; "I must attend to this matter"; "She took care of this business"	6.904
15. see -- (receive as a specified guest; "the doctor will see you now"; "The minister doesn't see anybody before noon"	6.666
16. see, watch -- (check, try, or ascertain; "See whether it works!"	6.428
17. go steady, go out, date, see -- (date regularly; have a steady relationship with; "Did you know that she is seeing her psychiatrist?" "He is dating his former wife again!"	6.190
18. see -- (see and understand, have a good eye; "The artist must first learn to see"	5.952
19. see -- (deliberate or decide; "See whether you can come tomorrow"	5.714
20. see, escort -- (accompany or escort: "I'll see you to the door"	5.476
21. see -- (match the bet of another player; in poker	5.238
MAN (noun)	
1. man, adult male -- (an adult male person (as opposed to a woman); "there were two women and six men on the bus"	١٠
2. serviceman, military man, man, military personnel -- (someone who serves in the armed forces; "two men stood sentry duty"	9.545
3. man -- (the generic use of the word to refer to any human being; "it was every man for himself"	9.090
4. world, human race, humanity, humankind, human beings, humans, mankind, man -- (all of the inhabitants of the earth; "all the world loves a lover"	8.636
5. homo, man, human being, human -- (any living or extinct member of the family Hominidae	8.181
6. man -- (a male subordinate; "the chief stationed two men outside the building"; "he awaited word from his man in Havana"	7.727
7. man -- (an adult male person who has a manly character (virile and courageous);"the army will make a man of you" competent	7.272
8. man -- ((informal) a male person who plays a significant role (husband or lover or boyfriend) in the life of a particular woman; "she takes good care of her man"	6.818
9. valet, valet de chambre, gentleman, gentleman's gentleman, man -- (a manservant who acts as a personal attendant to his employer; "Jeeves was Bertie Wooster's man"	6.363
10. Man, Isle of Man -- (one of the British Isles in the Irish Sea	5.909
11. man, piece -- (a small object used in playing certain board games; "he taught me to set up the men on the chess board"; "he sacrificed a piece to get a strategic advantage"	5.454
MOUNTAIN (noun)	
1. mountain, mount -- (a land mass that projects well above its surroundings; higher than a hill	10
TELESCOPE (noun)	
1. telescope, scope -- (an instrument that magnifies the image of distant objects)	10

Table 4.11 WordNet overview of stems in sentence 4

The prepositions *on* and *with* are represented in the object oriented lexicon as abstract classes. These classes will not produce objects; rather they contain abstract methods that give the appropriate semantics imposed by these prepositions. The following table shows the association value of the preposition *on mountain* with other stems.

ON	Boy	See	Man	Telescope
MOUNTAIN	1	0	1	0

Table 4.12 Association values for prepositional phrase *on mountain*

Following table shows the association values of *with telescope*.

WITH	Boy	See	Man	Mountain
TELESCOPE	1	1	1	1

Table 4.13 Association values for prepositional phrase *with telescope*

For *boy* all senses are possible. For *man* also all senses from 1 to 9 are possible. For simplicity we take the first sense only.

Discussion

This sentence shows the ambiguity caused by the presence of prepositions. In such cases, more than one meaning are equally likely as long as they are semantically consistent. Thus, OOSEMAN reports all possible interpretations. Prepositions *On* and *With* finds the association values with other stems in the sentence. These values are given in tables 4.12 & 4.13. These association values are used in the main process where the verb is associated with other stems as shown in table 4.14 in the next page.

	Boy (10)	Man (10)	On mountain (10)	With telescope (10)	Overall meaning
See (10)	10x10x1 = 100 (the boy saw)	10x10x1 = 100 (the man been seen)	Boy on mountain 10x10x1 = 100	Boy having telescope 10x10x1 = 100	The boy who was on the mountain and was having a telescope saw the man
				Man having telescope 10x10x1 = 100	The boy who was on the mountain saw a man who carried a telescope
				Seeing with telescope 10x10x1 = 100	The boy who was on the mountain saw through telescope the man
				Mountain with telescope 100	The boy who was on the mountain that has a telescope, saw the man
			Man on mountain 10x10x1 = 100	Boy having telescope 10x10x1 = 100	The boy who had a telescope saw the man who was on the mountain
				Man having telescope 10x10x1 = 100	The boy saw a man on the mountain and this man was carrying a telescope
				Seeing with telescope 10x10x1 = 100	The boy saw using a telescope a man who was on the mountain
				Mountain with telescope 100	The boy saw the man who was on the mountain that has a telescope

Table 4.14 Interpretations of sentence 4

CHAPTER 5

CONCLUSION

This thesis work discussed a novel system (OSEMAN) that based entirely on Object-Orientation concepts in fulfilling the requirements of Semantic Analysis of natural languages. After giving background information related to various issues in NLP and semantic analysis, OSEMAN was described in detail. The model was verified using some case study examples. This chapter discusses the contribution of this thesis work to the field of NLP. This is emphasized by discussing the advantages and limitations of OSEMAN. This chapter concludes by highlighting some future additions that could be incorporated into the model.

5.1 THESIS CONTRIBUTION

This thesis mainly contributed to the field of NLP in general and particularly to semantic analysis. The outcome of this thesis is the OSEMAN model. This model has many contributions to the field of semantic analysis of natural languages. OSEMAN incorporates object-orientation concepts in semantic analysis. This is achieved through object-oriented lexicon. Object-Orientation is the most contemporary paradigm and many applications have chosen this paradigm in implementation. OSEMAN is believed to be the first model that attempts to perform semantic analysis using object-orientation.

Another great contribution of the thesis is the object-oriented lexicon, which plays the major role behind the scene. The presence of an object-oriented lexicon that supports the model is regarded to be the greatest advantage of OSEMAN. The lexicon is assumed to be present. The difficult part in building the lexicon is the discovery of semantic class hierarchies of vocabulary words. However, relying on semantic hierarchies present in a lexical database like WordNet makes the effort of building the lexicon much less. Having the class hierarchy, the only thing needed is to define the classes by coding semantic and linguistic information of vocabulary words in terms of attributes and behaviors. The lexicon represents words as classes. Thus, adding new words are easy. The word to be added must be first properly classified semantically to know its appropriate position in the hierarchy. Only distinguishing attributes and methods that describe this word need to be considered. In this way, previous components are reused. The concept of encapsulation, where the details of implementation is hidden, makes it easy to change from one implementation of certain class methods to another. In summary, all plus points of object-orientation could be cites in favor of the lexicon.

Another contribution of this thesis is devising equation for finding rank values. This equation has a great impact on finding the combination value. The equation was made in a way such that higher frequently used sense of a word receives higher rank value.

Many existing NLP systems restrict their use in certain domain in order to resolve ambiguity. OSEMAN however, with the presence of the complete lexicon, does not restrict the text to be processed to belong to a specific domain. This will enable the system to be used in many real life applications.

Assigning weights and reporting all interpretations above a threshold is another contribution of the thesis and has a great impact in handling ambiguity. Many NLP systems report all possible interpretations of an ambiguous text without ranking these interpretations. As a result many inconsistent interpretations were considered. OSEMAN with the presence of weights ranks the possible interpretations and excludes all less 'promising' interpretations by the help of the threshold.

Constrains are another advantageous feature in OSEMAN. These constrains allow the processing to be governed by the requirements of the user. A user may wish to allow certain grammatical errors, or wants to restrict the processing to a certain domain.

Many research studies concentrate on building systems to handle only one aspect of semantic analysis. This is the case because of the difficulty and diversity of NLP. Chapter 2 described systems that handle only verbal polysemy [25], noun sequence [22] and prepositional phrase attachments [12]. Since OSEMAN relies on an object-oriented lexicon that captures semantic and linguistic information of words in terms of attributes and behaviors, this enabled the system to handle many problems that were otherwise handled by a separate system for each. As mentioned in chapter 3, OSEMAN handled the concept of verbal polysemy and word sense disambiguation by using rank values assigned by the object-oriented lexicon. Adjectives, adverbs and prepositions were handled by considering them as abstract classes. In summary, having this lexicon in hand many issues in NLP could be handled simultaneously and which otherwise required a separate system to handle each.

5.2 LIMITATIONS & FUTURE WORK

Though OSEMAN exhibits a great power in semantic analysis, it suffers from some shortcomings. It takes a great effort to build the lexicon. However, with the presence of lexicons where words are classified into semantic hierarchy as in WordNet, much less effort is needed. Attributes and methods should be carefully chosen such that they reflect all semantic properties of the word. The proper encoding of class definition has its impact when giving the binary association values at the time of analysis.

OSEMAN considers sentences where one main verb is present and it finds the meaning of this sentence by searching for different semantic roles associated with this verb. In case of more complicated sentences where more than one verb is present, OSEMAN needs to be extended.

The threshold plays an important role in controlling the number of interpretations reported. This number varies from one sentence to other given the same threshold, as was shown in Chapter 4. Thus, the choice of the threshold is left for the user, and it is his responsibility to choose a threshold value in a way that gives the most likely interpretations of the sentence.

OSEMAN gives the association value between two words in the range between 0 or 1 to indicate the strong ness of the semantic association between these two words. The value was decided based on the length of the common parent of the two words. This approach could be extended by finding the number of attributes that are common between the two words to be associated. This will cause the overall weight to be more fine-tuned.

The rank value equation was chosen in a way such that rank values are distributed in fixed interval among the senses of the word. These values are maintained between the max and min that are predefined. The equation could be improved in future in a way to give more precise values.

OSEMAN performs the semantic analysis of a given sentence. In order to be part of the whole NLP system, OSEMAN must be extended to include discourse and pragmatic features. Also, some of the semantic analysis problems need to be covered like noun sequences, verbal nominalization, degree concept and question and negation sentences. OSEMAN could also be extended to handle natural languages other than English. OSEMAN can also be extended to be a semantic checker of input text. In this case an inconsistent sentence is given and the system should suggest ways to correct the semantic errors.

REFERENCES

[1] Busemann, S. et al "Message Classification in the Call Center" In Proc. Of ANLP-2000, Seattle, WA, 2000

[2] Breck, E. et al "How to Evaluate your Question Answering System Every Day and Still Get Real Work Done" in Proc. Of 2nd International Conference on Language Resources and Evaluation (LREC 2000) 2000

[3] Radev, D. et al "Centroid-Based Summarization of Multiple Documents: Sentence Extraction, Utility-Based Evaluation, and User Studies" NAACL/ANLP Workshop on Automatic Summarization, Seattle, WA, 2000

[4] Henderson, J. "Exploiting Diversity for Natural Language Parsing" Ph D thesis, The John Hopkins University, August 1999

[5] Jurafsky, D. and Martin, J. SPEECH and LANGUAGE PROCESSING:An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition. Prentice-Hall, 2000

[6] Manning, C. & Schutze, H. Foundations of Statistical NLP. MIT Press 1999

[7] Stefanini, M. and Danazeau, Y. "TALISMAN: A Multi-Agent System for NLP", XII Brazilian Symposium on Artificial Intelligence, Wainer, Carvalho (eds), LNAI 991, Springer Verlag, p. 310-320.1995

[8] Allen, James. Natural Language Understanding. The Benjamin Cummings. 1987

[9] Green, B. B. and Rubin, G. M., "Automated Grammatical Tagging of English . Department of Linguistics, Brown University, Rhode Island. 1971

[10] Garsid, R. et al. The Computational Analysis of English. Longman, London. 1987.

[11] Kupiec, J. Robust Part-of-Speech Tagging Using a Hidden Markov Model. Computer Speech and Language. (6):225-242. 1992

[12] Franz, A. Automatic Ambiguity Resolution in Natural Language Processing. Lecture notes in computer Science. Springer. 1996.

[13] Bever, T. G. The Cognitive Basis for Linguistic Structures. In Hayes, J.R., editor, Cognition and the Development of Language, New York. John Wiley. 1970

[14] Katz, J. and Fodor, J. The Structure of a Semantic Theory. Language, (39):170-210. 1963.

[15] Li, H. and Abe, N. "Word Clustering and Disambiguation Based On Co-Occurrence Data", Proc. of COLING-ACL'98. 1998

[16] Hindle, D. and Rooth, M. Structural Ambiguity and Lexical Relations. Computational Linguistics, 19 (1): 103 – 120. 1993.

[17] Ford, M. et al. A Cometence-based Theory of Syntactic Closure. In The Mental Representation of Grammatical Relations, Cambridge, MA. MIT Press 1982.

[18] Resnik, P. and Hearst, M. Structural Ambiguity and Conceptual Relations. In Proceedings of the Workshop on Very Lange Corpora, pp: 58 – 64. 1993.

[19] Rich, Elaine & Kevin Knight. Artificial Intelligence. 2nd Ed. McGraw Hill. 1991.

[20] Li, Hang, A Probabilistic Approach to Lexical Semantic Knowledge Acquisition and Structural Disambiguation". Ph D dissertation. University of Tokyo, 1998

[21] Hull, R.D. and Fernando Gomez "Semantic Interpretation of Nominalization" Proc. Of AAAI, 1996

[22] Vanderwende,L. "Algorithm for Automatic Interpretation of Noun Sequences" Proc of COLING- 1994

[23] Riloff, E. and Shepherd, J. "A Corpus-Based Approach for Building Semantic Lexicon" In Proc. Of the 2nd Conference on Empirical Methods in Natural Language Processing. 1997

[24] Sang, E. "Noun Phrase Recogniton by System Combination" In Proc. Of NAACL 2000, Seattle, WA, 2000

[25] Fukumoto,F. and Jun'ichi Tsujii. "Automatic Recognition of Verbal Polysemy" In Proc. of COLING-94. 1994.

[26] Androutsopoulos, I. "Temporal Meaning Representations in a Natural Language Front-End". In Proc. Of the 12th International Symposium on Languages for intentional Programming, Athens, Greece. 1999

[27] Aref, M. "Object Orientation in Natural Languages Processing" Proc. Of the 13th International Conference on IEA/AIE 2000.

[28] Aref, M. "Object-Oriented Approach for Morphological Analysis" Proc. Of 15th National Computer Conference, Saudi Arabia, 1997.

[29] Aref, M and Abdul-Baqi, M. "*An Object-Oriented Approach for Semantic Analysis*" Proceedings of IC-AI 2000, Las Vegas, USA . July 2000

[30] Schacht, S. and Udo Hahn. "A denotational Semantics for Joining Description Logics and Object-Oriented Programming". In Proc. Of SCAI 97. The sixth Scandinavian Conference on AI, Finland, August 1997.

[31] Aref, M. "Bilingual Knowledge Representation" Proc. Of 6th International Conf. And Exhibition on Multi-lingual Computing. 1998.

[32] Yaeger, G. 'Operations Order NLP Assistant an Object Oriented Knowledge Extraction Tool'. In Proc. Of PACLP99, London, UK. 19-21 Apr. 1999

[33] K. Kaikhah, 'LOGONS: An Object Oriented Natural Language Generation System' *Proceedings of the Second International Workshop on Human Interface Technology, October '95, Japan*

[34] Miller et al "Introduction to WordNet: an On-line Lexical Database" Technical Paper: Princeton University, 1993

[35] Molla, A. et al. "A Real World Implementation of Answer Extraction". Proc. of 9th International Conference and Workshop on Database and Expert Systems. Workshop "Natural Language and Information Systems" (NLIS'98). Vienna: 1998

[36] Mueller, Eric T. "Prospects for in-depth story understanding by Computer" 1999. Technical Paper.
Available http://arxiv.org/html/cs/0003003

[37] Slocum, J. Machine Translation Systems. Cambridge: Cambridge University Press. 1988.

[38] Spark Jones, Karen. "Automatic summarizing: factor and derections". Advances in automatic text summarization . MIT press, 1998.

[39] Wallace, M. Communicating with Databases in Natural Languages. Ellis Horwood Series in AI. 1984

Vita

- Abdul-Baqi Muhammad Sharaf-al-islam
- Born in Rajshahi, Bangladesh on May 18, 1973.
- Received Bachelor of Science (BSc) degree in Computer Science from King Fahd University of Petroleum & Minerals, Dhahran in August 1997.
- Joined King Fahd University of Petroleum & Minerals in February 1998.
- Currently three Publications.
- E-mail: abdulbaqi@mailandnews.com
- Homepage: http://abdul-baqi.homepage.com