2001

An experiment in intelligent text processing.

Rayan. Raghuram
University of Windsor

An Experiment in Intelligent Text Processing

by
Rayan Raghuram

A Thesis
To be Submitted to the faculty of Graduate Studies and Research
Through the School of Computer Science
In Partial Fulfillment of the Requirements for
The Degree of Master of Science at the
University of Windsor

Windsor, Ontario
Canada
2001

Abstract

Traditional approaches to information retrieval (IR) are statistical in nature. These approaches depend on data gathered from corpus analysis. As much as such approaches provide for immediately practical computational models in IR, they exhibit a maximal accuracy of 40% when applied to open-ended corpus. Language-based approaches, on the other hand, provide for more intuitive solutions to text retrieval with higher precision accuracy. The problem remains in that, these approaches require natural language understanding (NLU), and we do not have language understanding as of yet. It is commonly accepted that we require vast amounts of commonsense knowledge to attempt problems in NLU. It is the lack of such proper/complete knowledge structures to represent commonsense knowledge, that has left intelligent IR a little short of being abandoned.

In this thesis, we emphasize that complete NLU is not necessary for intelligent IR, as we do not have to understand the text completely. It seems that discovering the **aboutness** of text is sufficient to perform intelligent IR with precision accuracy that is far better than the traditional approaches. We prove this thesis existentially by implementing Digital Agora: An intelligent IR system that indexes/retrieves text based on subject content. Although we were successful in identifying that intelligent IR based on conceptual analysis is possible without complete NLU, we observed that the rather inefficient computational complexity that such an approach demands for, makes it impractical. Identifying this complexity bottle-neck to that of lexical disambiguation, we implement, test and present initial results of a computational model based on the Formal Ontology that attempts parallel marker-propagation at lexical disambiguation.

Dedications

In fond memories of the authors aunt, Mrs.Yarlgada Jayalakshmi, who left this world on the 14th of April, 1997, he dedicates this work to his niece Miss. Jayani S Vundavalli, born on the 9th of May, 2001, who he believes has joined his family to bring them joy and happiness.

Acknowledgements

The work described in this report has been carried out at the School of Computer Science, University of Windsor. This thesis was completed under the guidance of the following committee:

Dr. Adrien V. D. Hoven (vdhoven@uwindsor.ca) - External Reader
Dr. Richard A. Frost (frost1@uwindsor.ca) - Internal Reader
Dr. Robert D. Kent (rkent@uwindsor.ca) - Advisor
Dr. Walid Saba (saba@uwindsor.ca) - Advisor

The author is thankful to his committee for their valuable support, guidance, and advice, without which this thesis would still remain a challenge.

The author is also grateful to Dr. Walid Saba, for going beyond his supervisory role, in introducing him to this line of work, instilling into him this flair for language, and giving him undivided attention over the last few years. Last, but not the least, apart from his parents, the author also expresses his gratitude to his grandmother, Mrs. M. Bullammai, and his cousin/role-model/mentor Mr. M.V.S. Chowdry for their love, and financial/moral support without which his educational career at the University of Windsor would never have been possible.

Contents

List of Figures

Chapter 1

Introduction

This thesis is concerned with the development of language-based text retrieval systems. While information retrieval (IR) has been an active field for several decades, the explosion of the world-wide web and the plethora of information has created an urgency for better tools to access, visualize and summarize information more intelligently. Traditional IR systems have been based on the Boolean model of key-word indexing. These systems perform with a precision accuracy of 40%, at best. For several years, researchers in natural language processing (NLP) have attempted to employ language processing techniques to improve the performance of IR systems.

In recent years however, NLP approaches have gained some prominence, and an active area of research has been the use of on-line semantic dictionaries, such as Wordnet (Miller et al., 1998), as well as more advanced NLP-based and knowledge-based knowledge bases (Lenat et al., 1995), (Mahesh, 1996). One of the main criticisms of

NLP-based approaches to IR has been that the technology has not matured enough to cope with large and open-ended corpus. While this is true, our thesis is that some form of language processing can be effectively used to improve the performance of IR systems, beyond the traditional approaches. The assumptions behind our thesis can be explained as follows: while full natural language understanding is still not practically attainable, IR does not actually require a full understanding (or comprehension) of the entire text. IR only requires us to discover the **aboutness** of the text. That much, we believe can be done using NLP techniques. Our thesis, therefore is the following:

> *"NLP approaches can be effectively used to discover the aboutness of large, open-ended textual sources".*

Our thesis is proved existentially, i.e., by actually building a system that supports our thesis. In this work we describe such an experiment. While the experiment we conducted does support our claim, we encountered some major limitations that we will discuss in some detail. To set the stage for our experiment, we must clarify two points. First, our work differs somewhat from the traditional work in IR in that it is not rooted in the traditional model of building an inverted list of key-words as an index. Instead, our approach is based on performing linguistic and conceptual analysis to "discover" the key topics of a document (i.e., what the document is really about). This approach necessarily involves the use of some semantic network to perform conceptual analysis and to perform lexical disambiguation. Thus, our review of the relevant literature tended to be work in NLP and knowledge representation more so than traditional (statistical, key-word based, etc.) IR work.

1.1 Role of AI/NLP in Coping with the Information Overload Problem

Since the early 50's, research in artificial intelligence (AI) has had a mixture of successes and failures. Under the banner of AI there have been many research areas that have been actively pursued, such as machine translation, automatic theorem proving, natural

language understanding (NLU), knowledge-based expert systems, decision support systems, planning, robotics, and so on. In the mid 80's, AI made many over ambitious promises that were not fulfilled. This resulted in a diminishing interest in AI (in both industry and academia), and although the field did not deliver as the pioneers predicted in the early 60's and 70's, AI was still alive behind the scenes throughout the 80's and 90's. During this phase most of the achievements in the field were in expert systems, and primarily in rule-based decision support systems. However, apart from the expert system that was scheduling security personnel in the 1994 winter Olympics in Lillehamer, Norway, built by Neuron Data of Palo Alto, California or Microsoft's interactive CD's, Bob and Liz, it had been a decade since AI had taken the spotlight on the news (Hedberg, 1995). In the late 90's, although industry viewed this field as a pool of techniques that could be embedded in existing software products, researchers had a totally different perspective. AI was far more than just rule-based systems and case-based reasoning; AI was an integration of technologies that had matured.

In fact, most major corporations (e.g. Unisys, IBM, AT&T, Cannon & Xerox) had moved problems of NLU/NLP, language generation, human-computer interaction, vision, speech recognition and machine learning to top priorities on their agenda. Furthermore, the explosion of the web and the trend of consumer-based computing had created a urgent need for intelligent software that could communicate with users in natural language, and could help them search and navigate a globally inter-connected information jungle. Ironically, the recent interest in this field has only made researchers more cautious about their claims. That is, while AI in the 80's was more hype than reality, what we seem to have today is more science and advanced techniques than hype. In fact, many success stories of late are not even being advertised as AI success stories, but often go under such labels as "advanced" or "emerging" technologies, or "intelligent systems".

Regardless of it's history, there remain many challenges and problems to be tackled in AI, not least of which is natural language understanding.

1.2 Natural Language Understanding

Although quite different, natural language understanding (NLU) and natural language processing (NLP) are often (wrongly) used interchangeably. While NLP encompasses any language processing activity, NLU is concerned with understanding (comprehension) natural-language utterances, a process that must inevitably involve very complex forms of commonsense reasoning. To this end, the commonsense knowledge-representation acts as the main resource lexicon, and it plays a crucial role in many of the critical functionality of an NLU system, for example in tasks such as lexical disambiguation, anaphora resolution, the resolution of quantifier scope ambiguities, etc., as will be discussed later in this report.

The primary focus of this report is to understand the various complications involved in understanding language, describe the nature of commonsense knowledge relevant to language understanding, survey existing knowledge-repositories that represent such forms of knowledge, identify drawbacks in these knowledge structures, and suggest a computational model that proposes parallel marker-propagation as a reasoning strategy to attempt lexical ambiguity.

1.3 Commonsense knowledge

Although many difficult problems in specialized fields have been solved using computers, no program to date can do commonsense reasoning, one of the simplest things that even a child can do (Minsky, 2000).

Commonsense knowledge is information that every human knows regardless of his age, sex, education, creed or nationality. It is the kind of knowledge that is implicitly assumed when making inferences in language understanding or in performing some activity that requires some form of reasoning. According to (Lenat, 1998), commonsense knowledge constitutes those bits of information that we never express in a conversation out of fear of

embarrassment. Things like never hold the coffee cup bottom up when drinking coffee from it, or if you throw a ball in the air, expect it to start falling down and so on. As trivial as they may seem, such forms of information are necessary to understand the simplest of sentences in language.

1.4 The Structure of this Report

As stated above, our approach is based on using NLP and KR techniques to improve the performance of IR systems. Therefore, we first present a somewhat detailed review of some work in NLU and knowledge representation. As the experiment differs from traditional statistical approaches to information retrieval, in that it is based on semantic interpretation to discover the aboutness of information, we dedicate the first few chapters to understand the kind of a computational model that such an approach calls for. In brief, we set the stage by surveying ontologies, knowledge-representations, semantic networks, meaning representations and reasoning strategies for semantic networks.

Apart from an overview of the common general-purpose commonsense knowledge-representations used in NLP like Wordnet, Cyc, and so on, we identify their limitations in supporting language based intelligent text retrieval. We then explain the process that went into building Digital Agora: An intelligent text-retrieval system that indexes/retrieves documents based on topics (subjects). Apart from discussing how the precision accuracy of searches is enhanced by such an intelligent text retrieval system, we identify the effectiveness (rather high complexity) of the involved process as a major limitation its practical acceptability. We further narrow down this complexity bottle-neck to identifying its root as that of lexical disambiguation, caused due to the lack of well-defined/complete semantic structures that contain the relevant knowledge to perform lexical disambiguation on open-ended corpus. Finally, we discuss initial results of an effective model for lexical disambiguation as a solution to overcoming this complexity bottle-neck.

In chapter 2, we overview language understanding as a process that involves the resolution of various ambiguities that prevail in language. Apart from emphasizing the need for commonsense knowledge in NLU/NLP, we identify the various ambiguities that occur in language to discuss how commonsense reasoning can aid in the resolution of these ambiguities. Lexical disambiguation, anaphora resolution, prepositional-phrase attachments, quantifier-scope ambiguities, plan recognition and scenario recognition are introduced as some of the immediate problems in NLU that require such forms of commonsense knowledge.

In chapter 3, we probe to understand knowledge representation and reasoning (KRR) as a prominent field of research in AI. Apart from an overview of the important features of KRR, we discuss semantic networks as appropriate KR mechanisms. We then consider the task of defining what an Ontology/ontology is. Although Ontology, ontology and knowledge representation are used interchangeably, they have significant differences. We discuss these differences and shed light on the various relations that hold between them. Apart from introducing some relevant terminology, we also talk about the lexical matrix to show the interconnectedness between concepts and lexemes. We also discuss default reasoning and spreading activation as reasoning strategies that are applicable to semantic networks. Apart from elaborating on inheritance and reduction in semantic networks, we illustrate the importance of default reasoning when in relevance to representations of commonsense knowledge. Finally, we identify spreading activation as an appropriate reasoning strategy for NLP tasks that use commonsense knowledge, and discuss variations of spreading activation commonly used in such applications.

Chapter 4 is an overview of existing semantic networks, ontologies and knowledge bases commonly used in NLU. Apart from discussing various representations used in these structures, we analyze their pro's and con's. Cyc, Wordnet and Mikrokosmos, being the most commonly used ontologies in NLU, we study them in greater detail. Cyc is an ongoing commercial effort that was started in 1984, whereas Mikrokosmos and Wordnet

originate from the academia. We study each of these ontologies, and identify multiple inheritance as a common drawback prevalent in all of them.

In chapter 5 we describe Digital Agora, an intelligent text retrieval system that aims to index/retrieve information based on subject-content. Unlike traditional key-word/key-concept based information retrieval, we illustrate the syntactic/semantic/pragmatic computation that such an approach entails. Apart from describing the underlying model and design of Digital Agora (one that is significantly different from traditional IR models), we present snippets of code to better illustrate the underlying functionality. Although Digital Agora is not based on complete language understanding (NLU), lexical ambiguity and nomainal decomposition need to be addressed by the system to facilitate topic-based intelligent text retrieval. Although the focus of this chapter is to describe the system, as a continuous them throughout this chapter, we shed light upon various heuristics and tradeoffs considered during its implementation. Finally, showing how such an intelligent information retrieval system improves search precision, we identify the underlying computational complexity of the system as a limitation to its acceptability as a practical solution. Tracing this high complexity down to lexical disambiguation, we identify the roots of the problem to that of inconsistency and incompleteness of the knowledge represented in these semantic networks and KRs.

In chapter 6, we re-visit lexical ambiguity with more rigor. The focus here is to suggest a computational model to approach this problem. Discussing traditional attempts at resolving this semantic ambiguity, and identifying their limitations, we suggest the use of selectional-restrictions as a competent and efficient solution to this problem. Identifying the necessary commonsense knowledge to attempt lexical disambiguation, we describe how this knowledge can be integrated on top of the Wordnet semantic network. Identifying that this upper-level extension is a consequence of a formal ontology, we suggest using various cues in language to discover this structure. Keeping in mind that deriving such an ontology is a tedious long-term venture (as with the case of Cyc, Wordnet, etc..), we simulate a portion of this structure on top of Wordnet in an effort to

implement the proposed solution. Moreover, we exploit the asynchronous parallelism that this model caters for, by implementing and testing a parallel marker-propagation algorithm on this simulation.

Lastly, in the conclusion, we summarize the important observations of this report, and discuss future extensions to our research. Briefly summarizing the experiment, we shed light upon other critical areas of Digital Agora where there remains room for modification to improve its effectiveness and efficiency.

Chapter 2

Overview of Natural Language Understanding (NLU)

Most corporations have begun to see electronic trade as the prospective future. Thousands of small software companies around the world are focused on improving customer satisfaction by making software interfaces more and more user friendly. Human-interface dialog, suggestive selling and handling customer concerns are some of the efforts in this direction. The inability of computers to understand language hinders any progress along these lines. The various unsolved ambiguities in English leave us no option but to compensate on trying to understand language in it's fullest. A precision of 40% is the best we have achieved in processing language. To overcome these ambiguous situations and make sense of the simplest sentences, we have to equip our software with commonsense knowledge and the ability to reason with it. Unless these barriers are broken, the dream of a complete electronic market will never come true. Unless the content available to us is understood, there is absolutely no hope for any higher level activity. This lack of NLU has left us no choice but to make many comprises. Although,

these comprises may prove comparatively insignificant when considered in isolation, for the completion of a specific task, they scale up to a large deficiency when trying to instill structure to the large ocean of ad-hoc information that we call the web. In this section, we discuss some the most important hindrances to a complete NLU and how they can be resolved using common sense knowledge.

2.1 Lexical Disambiguation

Lexical disambiguation is one of the preliminary problems in NLU/NLP. The roots of this problem lie in the fact that words in the English lexicon can have more than one meaning. In fact, most words in the English language are ambiguous, i.e. mean different things when used in different contexts. To complicate things further, most conceptual things can be represented by more than one word, thus making the relationship between words and meanings many-to-many. The notion of disambiguating a word by looking at a small window around it was apparently suggested by Warren Weaver in 1955, in the context of machine translation (Jurfsaky et al., 2000). To familiarize ourselves with some terminology used in this stream of NLU/NLP, the **sense** of a word is a possible meaning that a word can represent, whereas a **synonym** of a concept is a possible word that can represent the concept. The **polysymy** of a word, on the other hand, is the total count of senses that the word can possibly represent. In our use of English, we humans seem to have no problem in deciphering the correct sense of a given word when it is used in a sentence. This identification of the appropriate sense of a word based on where and how it is used is known as lexical disambiguation. To describe the difficulty of the problem, consider a commonly used word like star. According to the Wordnet semantic network, star has seven meanings. Described below are each of the senses along with their synonyms and definitions.

1) **star:** (Astronomy) A celestial body of hot gases that radiates energy from thermonuclear reactions in the interior.
2) **star, ace, adept, sensation, maven, virtuoso, genius, hotshot, whiz, whizz, wizard, wiz:** Someone who is highly skilled.
3) **star:** Any celestial body visible (as a point of light) from the Earth at night.

4) **star, principal, lead:** An actor who plays a principal role.
5) **star:** A plane figure with 5 or more points; often used as an emblem.
6) **star, headliner:** A performer who receives prominent billing.
7) **star, asterisk:** A star-shaped character * used in printing.

Similarly, most words used in the common vocabulary are highly polysemous. Therefore, to do any form of NLU/NLP it becomes a necessity to disambiguate the right meaning of the word. Most approaches to disambiguation use the context (other words around the word) to help identify the right meaning. Statistics of contextual occurrence and similarity measures between meanings are two of the many approaches used in lexical disambiguation. Although in a closed test set, a 90% accuracy in disambiguation has been claimed, results have been far less promising in open test sets (a maximum precision accuracy of 40%).

8) The moon is a star that reflects light.

To explain the process briefly, in order to disambiguate **star** in sentence (8) under the context set **{moon, reflect, light}**, each sense of **star** is matched to every word in the context. This match is usually some form of a similarity measure based on heuristics and rules. The sense with the maximum similarity to the context set is then selected. In this case the words **moon** and **light** would create a bias towards selecting either meaning (1) or meaning (3) which are in fact the most appropriate in this context. In the above example we were fortunate to have a good context set to help disambiguate correctly, but this is not always the case, as shown in sentence (9).

9) The astronaut is engaged to a star.

In sentence (9), the word **astronaut** in the context set causes a shift towards either meanings (1) or (3), whereas the intended meanings are either (4) or (6). These forms of sentences are not rare occurrences in the usage of English and therefore should not be trivialized. To stretch the existing accuracy of disambiguation techniques any further we need commonsense knowledge. If in the case of sentence (9), commonsense knowledge

told us that people get engaged to people and therefore a star in this case should be a person, the correct meanings would have been chosen. Fig.1 illustrates one possible representation of such commonsense knowledge.

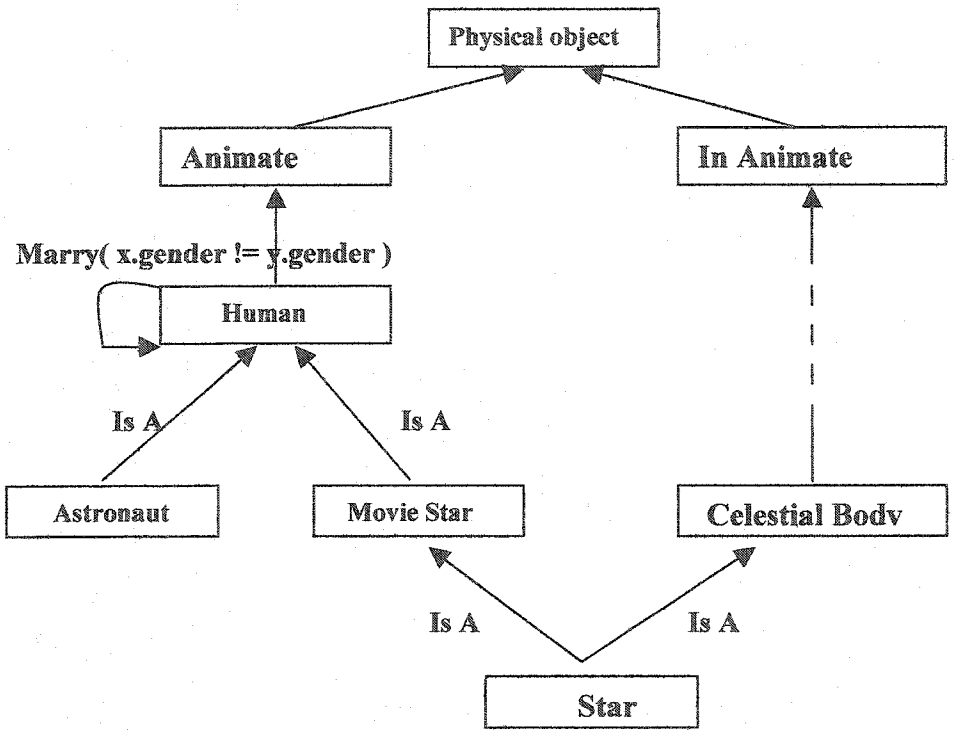


Fig 1: A conceptualization of relations between concepts.

2.2 Anaphora Resolution

Anaphora resolution, also commonly known as reference resolution or pronoun resolution is a problem that deals with the reference of pronouns to objects that have been mentioned previously.

1) John was hungry.
2) He ate in a chinese restaurant.

The problem of reference here is to decide on the object that **he** in sentence (2) refers to. In this case one may argue that the decision is trivial because the syntactic parse of

sentence (1) reveals that **John** is the object of the sentence and therefore **he** from sentence (2) refers to **John**. Syntax of the sentences does not always reveal enough information to resolve references as depicted in sentences (3) and (4).

3) Gallielo and Jacobson discovered the moon and mars.
4) They orbit around the earth.

In the case of (3) and (4), neither syntactic nor semantic information provides adequate information about whether **they** in sentence (4) refers to **Gallielo & Jacobson** or **moon & mars**. To solve this form of an ambiguity we need some form of a pragmatic information. In this case, commonsense knowledge that **Gallelio & Jacobson** are people, and people do not orbit the earth will help us in deciding the **they** in sentence (4) refers to **moon & mars**. If sentence (4) is modified to sentence (5) :-

5) They won the Discoverer's award.

All of sudden **they** in sentence (5) refers to **Gallileo &Jacobson** and commonsense knowledge that **moon & mars** do not win awards helps in this choice. Let us consider sentences (6) and (7) to observe the need for commonsense knowledge from a slightly different perspective.

2) After John proposed to Marsha, they found a group of preachers to get married.
3) For the honeymoon, they went to Hawaii.

In sentence (7), knowledge about what **John & Marsha** can and cannot do is not enough to solve the ambiguity because even the **group of preachers** can go on a honeymoon. Therefore, it might be appropriate to use a commonsense heuristic that honeymoons usually go hand in hand with marriages to decide that **they** refers to **John & Marsha** and not to the **group of preachers**.

4) John shot a soldier. He immediately fell down.
5) John shot a soldier. He immediately fled away.

In sentences (8) and (9), it is interesting to note that although the syntactic structure of the two sentences is exactly the same, the change of verbs in the second half from **fell** to **fled** causes a shift in the reference pointed to by **he** from **soldier** to **John**. This is so in sentence (9) because we know from common sense that if a person shoots somebody (especially an official), he tends to flee out of fright whereas in sentence (8) one would assume the soldier fell down because if a person is shot, he tends to fall down from the impact. Fig 2 gives us a conceptual view of how this form of information might be represented in the knowledge base.

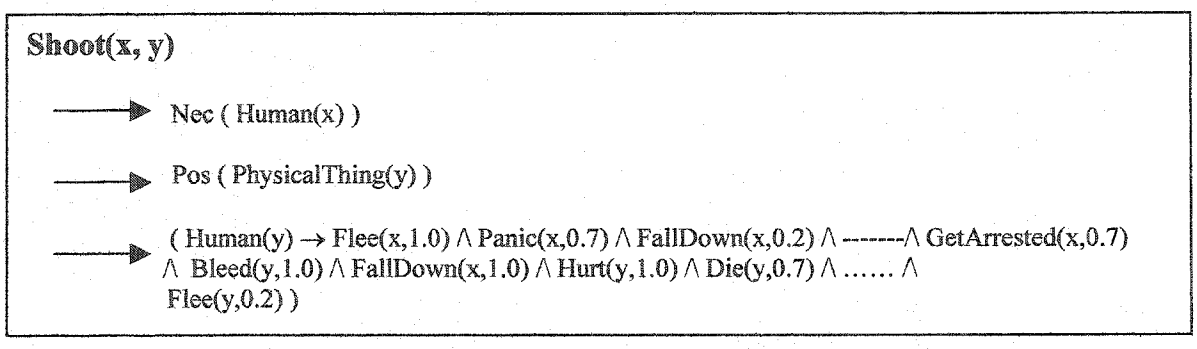


Fig 2: A conceptualization of information in the knowledge base

The representation in fig.2 can be read as follows. It is necessary for **x** to be a human being because only humans can shoot, and **y** can possibly be any physical thing, because we can shoot at any physical object. If **y** is a human, then we can attach a probability measure to the precedent of the implication; **x** flees, panics, falls down, etc. Similarly **y** bleeds, falls down, and gets hurt, and so on. Most inferences about the agent and the objects of the verb can be made in this form.

2.3 Prepositional Phrase Attachments

Prepositional phrases are formed by function words that combine nouns, pronouns or noun phrases like **the boy with the big nose, the city of joy** etc. Most times sentences in English are formed by attaching such phrases to complete sentences. This is an ingenious way of compacting into the same sentence (1) additional meaning that we wish to

represent which saves us the trouble of following the sentence (2) by an additional sentence (3).

1) John slept at the house on stilts.
2) John slept at the house.
3) The house is on stilts.

Although this is a very smart technique of compaction, it creates an ambiguity when trying to decipher it. Primarily the ambiguity in concern here is making the choice between whether **John slept on stilts at the house** or **John slept at the house on stilts.** In this example, it might be convenient to go with the direct reading, but in sentence (4) this heuristic does not apply.

4) John rode at the race on a Honda.

The correct interpretation of sentence (4) would be that **John rode on a Honda at the race**. To solve such an ambiguous situation one would depend on his/her commonsense knowledge of the world. The fact that John is a person, and people do not sleep on stilts guides to interpret sentence (1) in it's right sense, and similarly, the fact that people do not ride races helps us interpret (4) as **John rode a Honda at the race**. Another problematic situation that occurs out of prepositional attachments is in sentences similar to (5) and (6).

5) John takes tea with lemon.
6) John takes tea with cake.

The true understanding of (5) is that John drinks tea with lemon in it, whereas in (6) we mean that John eats cake along with his tea. Once again commonsense would tell us that people usually mix lemon in their tea to add to the taste, whereas no normal person would mix cake in his tea. Although the differences in each of these individual cases may seem minute, these small discrepancies build up exponentially to result in a total misinterpretation of the message being conveyed.

2.4 Quantifier Scope Ambiguities

Any sentence with **n** quantifiers has potentially **n!** possible readings that are due only to the different quantifier scope orderings. For example, (1) has potentially two different meanings, one implying a single house (wide scope 'a') and one implying several houses (narrow scope 'a').

1) John visited/advertised **a** house on **every** street.

Clearly, if the verb in sentence (1) was visited, then the most plausible reading would be the one implying that **On every street, John visited some house** since it is not plausible that it was the same house that was visited on every street; while it is quite plausible for John to advertise the **same** house on every street. Note that the determination of the most plausible scope ordering must involve some processes that are beyond syntax and semantics, but are in fact pragmatic considerations that can only be dealt by an inferencing module that must resort to some form of background knowledge.

Traditional approaches treated this problem by translating the ambiguity into a scope neutral form. After which these ambiguities were resolved in an incremental fashion by applying some preference rules, which were for most part syntactic in nature (Allen, 1984), (Alshawi et al., 1991), (Hobbs et al., 1987). There are several problems with this approach that are summarized in (Saba, 1999), (Saba, 2000). Briefly, these approaches suffer from a combinatorial explosion puzzle; it is neither cognitively nor computationally plausible that readers of ordinary English process **n!** alternative readings, applying various preference rules in an incremental fashion. Secondly, although some important results have been achieved with this approach, the problem remained largely unresolved in that only ad-hoc decisions were employed at the end to select the most plausible reading. Finally, these approaches do not model individual preferences in selecting the most plausible scope ordering, which actually seems to be the case. (Kutzman et al., 1992).

An inferencing module that seems to provide an answer to these problems and resolve scope ambiguities at the pragmatic level has been recently suggested in (Saba, 1999) and (Saba, 2000). The inferencing module suggested uses a quantification constraint (QC), which is a numerical constraint that is defined for every triple of a (binary) relation and two concepts. A QC, which is a piece of commonsense knowledge that we assume speakers of ordinary language compute in appropriately defined contexts, is assumed to reflect some individual's belief as to how some relation can-be/is-typically/must-be manifested at some (or at various) points in time. For example QC(on, house, street) = <many, 1> reflects a belief that a house cannot be on more than one street, but that on a street there could be many houses. Clearly such QCs must be a function of some temporal and modal aspects as well as a possible world, since a QC can change with different modalities, at different times, and in different possible worlds.

While the identification of the "relevant commonsense data" was important, the challenge in most of these problems is to also identify the proper inferencing algorithms that can utilize this knowledge.

2.5 Plan Recognition

Unlike any of the problems described above, plan recognition is not a form of ambiguity, but is more of discovering hidden truths. In layman's terms one might refer to plan recognition as reading between the lines. In any form of communication we use, some trivial details (i.e: common sense facts) are assumed and therefore never stated. For us as humans this is never a problem because we have the capability to fill in these blanks to make sense of the message in concern, but we have to understand that this is not the case with computers. To better explain what we mean, let us look at an example.

1) John wanted to kill himself.
2) He was looking for a rope.

If sentences (1) and (2) constituted the message, the possible truth that **John wanted to hang himself** or **John wanted to strangle himself** is hidden. In order to justify our using the term "possible truth" rather than "truth" in the former sentence, these form of inferences more often than not deal with the plausibility much rather than the certainty of the assumption. For example, John could be looking for a rope to do something totally different other than to hang himself, and could in reality want to kill himself with a gun. It is important to note that although these are only plausiblities, and these assumptions may not be truths after all, they are an integral part of the information required to understand the message in it's fullest, to thereby enable any further reasoning. More so, even we as human's would trust the plausibility and disregard the certainty to completely understand the message. We hope that there is no more need to explicitly state that we need some form of common sense knowledge to reason in this manner, because, it is a plausible truth that by now the reader should be able infer the hidden. A snap shot of where and how this form of knowledge will be represented in KB (Knowledge base) is shown below.

$(\forall x)$ (WantToDo(x,R) $\rightarrow$ R(x)

$(\forall x, y)$ (Kill(x,y) $\rightarrow$ Shot(x, y) $\vee$ Beat(x, y) $\vee$ ..... $\vee$ Hang(x, y))

$(\forall x, y)$ (Hang(x,y) $\rightarrow$ Used(x, rope) $\vee$ ..... $\vee$ Used(x, belt))

$(\forall x)$ (Kill(x,x) $\rightarrow$ CommitSuicide(x)

Fig 3: A snap shot of hidden plans.

To understand the set of implications in fig.3, a person **x** can kill a living thing **y** by either shooting **y**, beating **y** or hanging **y**. Similarly in the third implication **x** can hang **y** either with a rope or a belt. In the set of sentences we encountered the words **kill** and **rope**, now to make the inference all we are left to do is find the connecting predicate **hang** as shown in fig 3. From the last implication we can further conclude that, John wanted to commit suicide, because both arguments in the predicate Kill are the same.

2.6 Compound Nominals

Compound nominals are simple noun sequences that can be represented by Noun*, i.e: one or more nouns following each other. A simple set of context-free grammar rules like those shown below are sufficient for the production of these compounds.

Nominal ⟶ Noun
⟶ Noun Nominal

Since there are infinitely many compound nominals that can be generated in English by applying these production rules, they cannot be hard-coded into the lexicon. Therefore, the only way to derive the meaning of such compounds is by analyzing their constituents. The primary challenge is in that of discovering the implicit relation between the constituents. Listed below are three nominals and the respective implicit relations that are hidden in the compound. In each case, the relation is underlined. Although it is an acceptable heuristic to treat the last noun in the sequence as the head noun and view all nouns that precede it as its modifiers, identifying the correct relation in each case needs commonsense reasoning.

Animal-doctor - A doctor who treats animals.
Book-store – A store that sells books.
Knowledge-base – A repository that stores knowledge.

Although both syntactically and semantically there is no objection to deriving compound nominals using the set of production rules, some derivations may not be plausible (meaningful). Examples of such cases are derivations like **water sky**, **leaf bear** or **building fruit**. These derivations can be derived from the set of production rules mentioned above and their meanings can also be computed using semantic formalisms we choose, but they are not plausible and should therefore be avoided. This is an issue of pragmatics that requires commonsense. Under the assumption that humans generate only pragmatically plausible compounds, it is not a very important issue to be handled in

understanding nominals but plays more of a critical role while generating them as in the case of language generation.

2.7 Scenario Recognition

A scene is defined as a set of sub-events that are necessary for an action or event to take place. In our everyday lives when we communicate with each other to express ourselves, this information is never explicitly expressed by the sender of the message, but is rather taken for granted by the receiver. According to Lenat, this is the kind of knowledge, which we usually don't even tell children because it's so obvious. But that's knowledge that computer programs don't have, and until we build it into our systems, computer reasoning will still remain a challenge. Considering a possible message **I drove from Windsor to Ottawa yesterday**. Necessary sub-events that are implicitly assumed include:-

I drove a vehicle from Windsor to Ottawa.
I stopped at-least once to fill gas.
I made some form of payment for the gas.
I was on the road for at-least 5 hours.
:
:

Similarly, in the message **John drank a cup of coffee** necessary sub-events include: -

John had to hold the cup to be able to drink from it.
The cup was held face-up, bottom-down while drinking the coffee.
John consumed the coffee through his mouth.
:
:

Although some of this knowledge can be generated using smart knowledge structures, heuristics and algorithms, there is no way around hard-coding most of them into the knowledge base. This is primarily the reason why Cycorp has put together a team of blue-collar philosophers, psychologists and logicians to feed information to the Cyc knowledge base. To give the reader a better understanding of the magnitude of this problem, a Cycorp forecast in 1990 revealed their ambition to assemble a knowledge base

on the order of 10^8 axioms. Fortunately for us very limited scenario recognition is sufficient for NLU tasks. In robotics, on the other hand, where systems have to plan future activities and in some cases even execute these plans, complete recognition of the scenario becomes necessary. We use the word "fortunately" simply because there is an open problem associated with scenario recognition. Given a set of scenarios, selecting relevant scenarios to enable a plan is an NP-Complete problem. The Frame problem was first identified by McCarthy in relevance to representing knowledge through frames, an ingenious approach suggested by him. Although he believes that circumscription (selection of relevant frames by the elimination of irrelevant ones) is a possible approach to overcome this hurdle, a formal solution is yet to be coined (McCarthy, 1987).

Chapter 3

Overview of Knowledge Representation & Reasoning (KRR)

3.1 Knowledge Representation

Knowledge representation, more commonly known as knowledge engineering, is a study of computer science involved in structuring knowledge to be accessed by intelligent software. Although knowledge representation is a fairly new subject, in the technical sense, it has been a part of other disciplines like linguistics and philosophy ever since the first millennium BC. There are claims that the first roots of knowledge representation date back to 350 B.C., to Panini's Shastric Sanskrit Grammatical theory which not only proposed a formal syntax and vocabulary for a general-purpose language, but also provided an analysis of it's semantics (Briggs, 1985). Dating back to Plato and Aristotle, the words **epistemology**, **knowledge** and **representation** have played a prominent role in the philosophical vocabulary. Plato shifted the focus of philosophy from the nature of

knowledge to the representation of knowledge. Aristotle had to coin a terminology to facilitate the representation of knowledge during his day. Apart from the terminology, he also defined the scope of logic, physics, biology, psychology, linguistics, literary criticism, ethics, meta physics, and political science (Sowa, 1994). In effect, Aristotle was concerned with the classification of various domains. This, today, has evolved into what we call ontology design and knowledge representation.

The next important landmark in the history of knowledge representation was the first semantic network created in the third century AD by a philosopher Porphyry. This semantic network was a tree that represented Aristotle's categories in a hierarchy. A representation of this semantic network is presented in fig.4: -

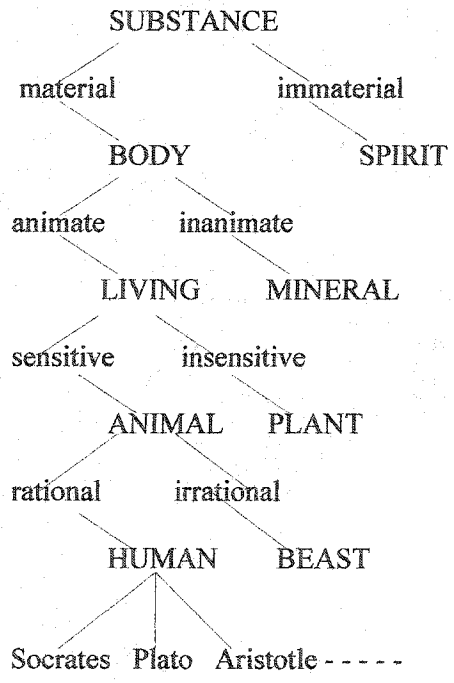


Fig 4: Tree of Porphyry, adapted from a Latin version by Calcagno (1952).

According to Aristotle (Sowa, 94), SUBSTANCE was the supreme genus (the most generic thing that existed in the world) for everything is a substance. BODY, SPIRIT, LIVING, MINERAL, ANIMAL and PLANT were subordinate genera, where as HUMAN and BEAST were species with Socrates, Plato and Aristotle as their instances.

The differentiation's in the genera were called differentiate which included material, immaterial, animate, inanimate, sensitive, insensitive, rational and irrational. Each category inherits all the differentiae along the path above it:-

- A rational sensitive animate material substance was a human.
- A sensitive animate material substance was an animal.
- An irrational sensitive animate material substance was a beast.
- An inanimate material substance was a mineral.

A more prominent landmark in the history of knowledge representation was the seventeenth century where the German philosopher Gottfried Willhelm Von Leibniz encoded this tree of Aristotle's categories as numbers and related them through computation. The primary goal of Leibinz's ingenious approach was to reduce reasoning to a computation. All differentiae were assigned a prime number, and all composite categories were derived by multiplying them. To illustrate this further let 2 be the prime number assigned to the supreme genus SUBSTANCE and any primes that follow be assigned to the differentiae :-

material = 3	immaterial = 5
animate = 7	inanimate = 11
sensitive = 13	insensitive = 17
rational = 19	irrational = 23

Then, every composite category can be represented as the product of all the differentiae that occur in the path above it. For example: -

- HUMAN would be represented by 2 * 3 * 7 * 13 * 19 = 10,374
- ANIMAL would be represented by 2 * 3 * 7 * 13 = 546
- BEAST would be represented by 2 * 3 * 7 * 13 * 23 = 12,558
- MINERAL would be represented by 2 * 3 * 11 = 66

Apart from representing these categories, Leibniz also defined operations to reason with them. To check for the "is a" relationship between two categories, the test was to determine that the child was perfectly divisible by the parent. For example a HUMAN is

an ANIMAL because 10,374 / 546 = 19. Although most of these ancient approaches had their limitations, it is intriguing to note that most of the prominent techniques used in knowledge representation to date originate from this early day when there was no computer and the only need to represent knowledge was to understand it better. These approaches to the classification of knowledge have a stronger bias towards "the correct classification" versus current approaches in AI, primarily because in AI, we classify knowledge to be able to reason with it, and not to understand it.

Moving from the ancient times to the modern day, as defined by Davis, Shrobe and Szolovits (Davis et al., 1993), knowledge representation as a discipline in AI is identified by five distinct roles that it plays, namely:

1) A Knowledge Representation is a surrogate.
2) A Knowledge Representation is a set of ontological commitments.
3) A Knowledge Representation is a fragmentary theory of intelligent reasoning.
4) A Knowledge Representation is a medium for efficient communication.
5) A Knowledge Representation is a Medium of human expression.

Fundamentally, a knowledge representation is a surrogate because it is a substitute for the real thing. It is a representation of some real entity that exists in the world around us and not the real thing in itself. We determine consequences by reasoning with these surrogates rather than taking action on the real entities. In this view even reasoning is also a surrogate because reasoning in itself is a process that is carried out internally although most things we reason about exist externally. Under this view, semantics is treated as the correspondence between the surrogate and it's referent in the real world. This is the basis for a strong claim, a room for error in reasoning will always exist because we deal only with surrogates and not with the real entities in themselves.

The second role is that of a set of ontological commitments that denotes the terms in which the world is being viewed. This notion arises from the fact that the world can be viewed across infinitely many dimensions. Although this role of a knowledge representation proves awfully important because it allows us to focus attention on those

aspects of the world that are most relevant to the domain specific problem at hand, it poses the inability to capture other aspects of the world that may be necessary. These ignored aspects may prove a barrier to reason both within the domain in question and across domains. More commonly known as the focusing-blurring effect, it is a prominent problem with domain specific knowledge bases that lack commonsense knowledge.

The next role of a knowledge representation is that of a fragmentary theory of intelligent reasoning which is derived from the motivation behind representing knowledge. From the perspective of a knowledge engineer, the primary motivation behind a knowledge representation is our ability to reason with it. In this role, a representation's theory can be made evident by (1) the representation's theory of fundamental conception of intelligent inference, (2) the set of inferences that the representation sanctions, and (3) the set of inferences that it recommends. A list of multiple views of intelligent reasoning and their intellectual origins are presented in (Davis et al., 1993). The forth role of a knowledge representation is as a medium for efficient computation. This builds on the mechanistic belief that reasoning is solely a computational process both in machines and in people. In this role, representing knowledge is not solely about reasoning but also about the computational efficiency associated with making the necessary inferences.

Lastly, the role of a knowledge representation as a medium of human expression, according to which representations are the means by which we express things about the world. Important questions to determine whether a knowledge representation satisfies this role include: How does it function as a medium of communication?, How easy is it to use?, How precise is it?, How general is it? Does it provide expressive adequacy? and so on. With a bias towards NLU/NLP, we assume that there exists a perfect match between the knowledge we have and the language we use to express it. With this perspective in mind, we give this role significant importance in our approach to representing knowledge, as illustrated in later the chapters.

Although these multiple roles of a knowledge representation may well be accepted in most disciplines like mathematical logic, biology, psychology, statistics and economics, a philosopher's perspective differs drastically, because in Philosophy, a knowledge representation is all about understanding the knowledge that exists around us and not about reasoning with it. In our opinion, specially so when dealing with common sense knowledge, we believe that this outlook to a knowledge representation is far superior because it does not have any bias towards any specific problem.

3.2 Ontology

The word "ontology" has recently gained immense popularity in AI. Although this word is used loosely in the knowledge engineering community, its meaning tends to be a bit vague. "Ontology" with capital 'O' denotes a philosophical discipline and ontology with a small 'o' denotes a specification of a conceptualization (Guarino et al., 1995). A conceptualization is an abstract view of the world that we intend to represent. In other words, a formal representation of knowledge is based on a conceptualization: the objects, concepts, and the other entities that are assumed to exist in some area of interest and the relationships that hold among them (Genesereth et al., 1987). In philosophy, Ontology is a systematic account for existence whereas in AI an ontology is a description (like a formal specification of a program) of the concepts and relationships that can exist for a community of agents or software that use the ontology (Gruber, 93). This definition of ontology coined by Gruber is the most widely accepted in the knowledge-engineering community.

Although there has been many a controversy about Gruber's definition, and other definitions have been proposed, but as a definition that is clear, precise and useful for the AI community, it still retains its popularity. This interpretation was coined from the perspective of knowledge sharing and reuse where knowledge is represented in a declarative formalism. The set of objects that can be represented is called the universe of discourse. All relationships between these objects are captured in the representational

vocabulary of these declarative formalisms. Therefore, the ontology of a program is a set of representational terms that can be used by the program. The entities in the universe of discourse are denoted by either classes, relations, functions or other objects. Apart from the objects themselves, constraints that restrict the well-formed use of these terms are necessary to the ontology. According to Enderton, Ontologies are often equated with taxonomic hierarchies of classes, class definitions and the subsumption relation, but ontology's need not be limited to these forms. Ontologies are also not limited to conservative definitions, that is, definitions in the traditional logic sense that only introduce terminology and do not add any knowledge about the world (Enderton, 1972). Ontology is a shared source of knowledge and vocabulary used by software and therefore guarantees only consistency and not completeness. We use common ontologies to describe ontological commitments for a set of agents so that they can communicate about a domain of discourse without necessarily operating on a globally shared theory. We say that an agent commits to an ontology if it's observable actions are consistent with the definitions of the ontology (Newell, 1982).

Other proposals made by the knowledge-sharing community to define ontology include ontology as an informal conceptual system, ontology as a formal semantic account, ontology as a representation of a conceptual system via a logical theory, and ontology as a meta-level specification of a logical theory (Guarino et al., 1995). Although each of these differ in how they interpret ontology, the common underlying outlook towards an onotology as an object rather than a discipline prevails. Five statements made by Guarino and Giaretta that make use of the term ontology explain in a nutshell the differences between an ontology & Ontology.

1) Ontological engineering is a branch of knowledge-engineering which uses Ontology to build ontologies.
2) An ontology is a special kind of knowledge base.
3) An ontology has its underlying conceptualization.
4) The same conceptualization may underlie many ontologies.
5) Two different knowledge bases may commit to the same ontology.

A different approach to better understand ontologies is to discuss what they contain. According to Fikes, an ontology contains a set of sentences (axioms) and a set of definitions. Definitions are expressions containing undefined primitive symbols. These symbols are given meaning by requiring that any interpretation of these symbols satisfies a given set of sentences in the conceptualization being represented. Any sentence that is not a tautology and is satisfied by the conceptualization can be included in the ontology (Fikes, 96). In this we understand that clearly ontologies are an integral part of knowledge-representation languages. Domain-specific languages include a domain-specific vocabulary and can therefore be considered to include an ontology. It is interesting to note that knowledge representation languages include functions or relations associated with the ontology. Examples of such relations include "instance of", "subclass of", "slot value type" and "slot cardinality". According to Sowa, logic is a pure form, and ontology provides the content. Without ontology, logic says nothing about anything. Without logic, ontology can only be discussed and represented in vague generalities (Sowa, 1995).

In our analysis, we were led to believe most AI research in ontologies is domain specific, and thereby elevates many important issues like reusability, integration and general standards to a greater importance. The goal towards an intelligent computer is that of possibly integrating one day all domain specific ontologies into one global ontology. The missing link to this venture is commonsense; knowledge that is not specific to any domain, and yet is applicable to every domain. Fortunately, many of the important issues associated with specific ontologies like reusability, integrity and standards can be ignored with the commonsense ontology, primarily because we need only one commonsense ontology to which all domain-specific ontologies can be attached. Formally known as a general ontology by the AI community, it poses one of the greatest barriers to intelligent computing. Two important categories that clearly distinguish a general-purpose ontology are listed below (Russell et al., 1995):

> *1) A general-purpose ontology should be applicable in more or less any special-purpose domain (with the addition of domain-specific axioms). This means that as far as possible, no representational issue can be finessed or brushed under the carpet. For example, a general ontology cannot use situation calculus, which finesses the issues of duration and simultaneity, because domains such as circuit timing analysis may require those issues to be handled properly.*

> *2) In any sufficiently demanding domain, different areas of knowledge must be unified because reasoning and problem solving may involve several areas simultaneously. A robot circuit repair system, for instance, needs to reason about circuits in terms of electrical connectivity and physical layout, and about time both for circuit-timing analysis and estimating labor costs. The sentences describing time therefore must be capable of being combined with those describing spatial layout, and must work equally well over various metrics like nano-seconds and minutes.*

However, the ultimate goal is to build a general ontology of commonsense knowledge. There is a considerable difference between a general ontology also known as a formal ontology, and a domain-specific ontology designed for a particular micro-world. The main difference is that knowledge encoded in a domain-specific ontology is not the kind that is assumed all speakers of ordinary language have recourse to (witness children's understanding of ordinary language, despite their lack of any domain-specific language). Fig.5 illustrates the boundaries of commonsense knowledge and domain-specific knowledge.

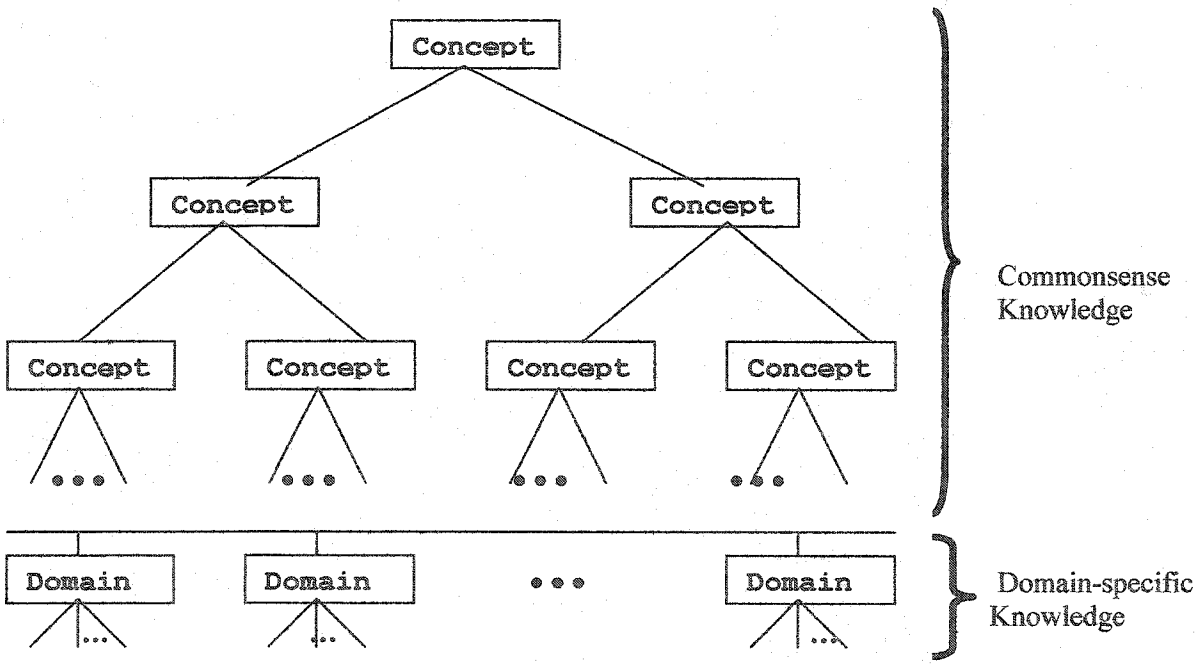


Fig 5: Commonsense knowledge vs domain-specific knowledge.

Instances of commonsense ontologies include the Cyc semantic network (Lenat et al., 1990), the Wordnet online lexical database (Miller et al., 1990). Pangloss (Knight et al., 1994) and Mikrokosmos (Mahesh et al., 1995). We stop with this basic description of as commonsense ontology for now, as we will venture to understand it in greater detail in later chapters of this report.

3.3 Ontology Vs Knowledge Representation

Although it is an accepted norm in the knowledge-engineering community to use the terms "knowledge-representation" and "ontology" interchangeably, there is a significant difference between the two. As mentioned in the previous sections, an ontology is a specification of a conceptualization and the later is a representation of this conceptualization in a form that computer programs can understand. This difference is well illustrated in the 2nd role where a knowledge representation is a set of ontological commitments (Davis et al., 1993). According to Davis, Shrobe and Szolovits, Ontology is the study of existence; the word comes from the Greek lexemes "ontos" [being] and "logos" [word]. For a database or knowledge base, ontology determines the categories of things that exist in an application domain. These categories represent the ontological commitments of the designer or the knowledge engineer. In other words, the designer or the knowledge engineer is constrained in how he/she can represent the categories in terms of the specified ontology. This in turn leads to constraints in how the represented knowledge can be interpreted by a user of the KB (person or software). In fact, this primary feature of a knowledge representation (KR) is the driving force behind it being a medium for efficient communication (role 4) and a medium of human expression (role 5). Ontologies are viewed as content theories whereas knowledge representation mechanisms like semantic nets, rule systems and frame languages are mechanism theories (Chandrasekaran et al., 1999). We use the terms "knowledge representation" and "ontology" in statements (1) to (6) below to understand them better.

1) Every knowledge-representation commits to an ontology.
2) More than one knowledge-representation may commit to the same ontology.

3) No knowledge-representation may commit to more than one ontology.
4) An ontology underlies a knowledge-representation.
5) A knowledge-representation does not exist without an ontology.
6) Reasoning with an ontology is not possible unless it is represented.

It is accepted in knowledge engineering that the choice of a knowledge-representation implicitly determines the way the world is viewed (ontology). This, therefore clearly determines the need for us to select a knowledge-representation that best describes the ontological commitments, and not vice versa. Although a knowledge-representation can be informally spoken of as a map of the ontological commitments to symbols, the complexity associated with the representation should not be trivialized. We share a quote from Lenat and Guha about their experience with Cyc (Lenat et al., 1990).

> *"The majority of work in knowledge representation has been concerned with the technicalities of relating predicate calculus to other formalisms, and with the details of various schemes for default reasoning. There has been almost an aversion to addressing problems that arise in actually representing large bodies of knowledge with content. The typical AI researcher seems to consider that task to be "just applications work". But there are deep, important issues that must be addressed if we are to have a large intelligent knowledge-based program: What ontological categories would make up an adequate set for carving up the universe? How are they related? What are the important things most humans know about solids? and so on."*

To enable all constraints and requirements of an ontology to be captured in reasoning, we use structured formal languages that can express knowledge in the representation process. Some examples of knowledge representation languages (KR languages) include propositional logic, first-order logics, temporal-logics, modal-logics and default logics. In many cases, we might extend existing formalisms to meet the needs of the ontology we are interested in representing. Examples of such languages include KL-ONE (Brachman et al., 1992), Cyc representation Language (Lenat et al., 1990) and SNeP- SLOG (Shapiro et al., 1981). Important features of a KR formalism are discussed in greater detail in the next. By far, formalisms like predicate calculus augmented with the "is a relation" have been used to represent ontologies. It is important to note that KR's are not the only entities that have ontology's underlying them. Whether consciously or not, we have been

using them long before we even found the need for knowledge representation in science. Other commonly used entities that commit to respective ontologies include mathematics, logic and programming languages. The ontology that underlies mathematics has the number system as its entities with specified relations amongst them and mathematical operations as the constrained operations permissible on them. Similarly Object-oriented design of software systems commit to the appropriate ontology. Objects, with their attributes represent the entities in the ontology, the inheritance relation models the relationship between objects and the procedures represent the permissible operations on these entities.

3.4 Meaning Representation and Common Sense Knowledge

To discuss meaning representation and commonsense knowledge in relevance to NLU, we analyze semantic networks from two perspectives. Firstly, from the perspective of how the knowledge (a collection of concepts) is represented in a semantic network and secondly how the represented concepts should be interpreted. In this venture, we hope to clarify the difference between a lexicon and a semantic network, and understand the notion of syntax, semantics and pragmatics as three distinct areas in language analysis. In a broad sense we associate a lexicon to syntactic information and semantic network to semantic information. Before we introduce common-sense knowledge into this picture, let us differentiate the roles and boundaries of lexicons and semantic networks in language analysis.

Ideally phonological, morphological and lexicological information should be included in a lexicon. Phonology deals with the sounds of a language. It provides information about the similarity of the sounds of various letters in a word, similarity of the sounds of different words, and the variance of sounds of the same word under different contexts. Phonology is an important feature for computer systems that deal with speech recognition, but is not relevant when dealing with text. Similarly, if the text is handwritten, graphology is used a substitute to phonology. Since in this survey we

assume printed text in electronic format, we can disregard both of the above mentioned. Morphology, on the other hand, deals with information about suffixes, prefixes, transformations between natural words to their inflected forms, and vice versa. The derivation of **magic** from **magical** and **school** from **pre-school** are examples of such transformations. Morphology also deals with the classification of words based on the part of speech that they represent. The five categories of words in the English lexicon are nouns, verbs, adjectives, adverbs and function words (Miller et al., 1990). There is the ambiguity of the same word falling in multiple categories, which will be discussed later in the section when we talk about how we interpret the lexicon. Lexicology goes beyond morphology to study the various meanings of a word which give rise to yet another ambiguity called lexical ambiguity.

The semantic network, on the other hand, focuses primarily on the various meanings that can be represented by a language. The conceptualization behind a lexicon is language dependant, where as in instances of a semantic network it is language neutral. Although a semantic network may be represented using a particular language, it should not be conceived as language-dependant, because the same semantic network can be represented using another language with minimal modifications. Whether or not a semantic network is a true replica of the structure we humans use to represent the concepts (entities) around us has been a topic of rigorous debate amongst researchers. As in the case of the ability of a word to represent many meanings, a meaning may also be represented by more than one word. This ambiguity of many words representing the same meaning is more of an aid than a problem to language analysis, as it provides a unique way to label concepts. Every concept in the semantic network can be labeled by its synset, the set of all words (synonyms) that can possibly represent the concept. In practice however, most semantic networks tend to prefer an indexing scheme over the synset in labeling concepts to facilitate space and time efficiency. The many to many map between words and concepts is best illustrated in a lexical matrix as in fig.6:

The lexical matrix is a conceptual graphical representation of the map between words and concepts. The rows of the matrix represent concepts and columns denote the words of the

vocabulary. We avoid restricting this map to the "English vocabulary" because this conceptualization is language independent and therefore applies to any natural language that supports ambiguity. The two dimensional matrix captures the notion that the same word can represent many concepts and similarly a concept can be represented by more than one word. An entry in a cell states that the column word represents the row meaning. To consider an example of this illustration, the label "$E_{1,1}$" identifies that the word "F_1" represents the meaning "M_1". All the words that represent a concept are called synonyms and form the synset for the concept. For example, the words "F_1" and "F_2" are synonyms and form the synset for the concept "M_1". A word is said to be polysemous (ambiguous) if represents more than one meaning. The total number of meanings that a word can represent is called the polysymy of the word. In the example illustrated in fig.6, "F_2" is the only polysymous word with a polysymy of 2. An example of a highly polysymous word is "break" which has 63 meanings as a verb according to the Wordnet semantic network.

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

Fig 6: The lexical matrix

Apart from the classification of concepts, the semantic network also captures relationships between various concepts. Some of these commonly used conceptual relations include the "is a" relation, the "part of" relation and the "substance of" relation. Concepts represent the nodes of the semantic network and the relations represent links

between these nodes. This capability to capture relationships between concepts provides a commitment for its users. In a broader sense, these hidden relationships describe the world model conceptualized in the semantic network.

Yet another important role of a semantic network, is that of a medium for representing concepts. A meaning can be represented in terms of its synset and the various semantic relations associated with it. A definition that defines the meaning is also included in its representation. For example in the Wordnet semantic network, a meaning is represented by its definition, synset, hypernyms, hyponyms, antonyms, coordinates and meronyms. Hyponomy and hypernomy (variously called subordination/superordination, subset/superset, or the "is a" relation) are semantic relations between word meanings. A concept represented by the synset {X,X',.....} is said to be a hyponym of the concept represented by the synset {Y, Y',.....}, if native speakers of English accept sentences constructed from such frames as *An X is (a kind of) Y*. The relation can be represented by including in {X,X',.....} a pointer to its superordinate, and including in {Y,Y',.....} pointers to its hyponyms (Miller et al., 1990). Hyponymy and hypernomy are transitive and asymmetrical relations (Lyons, 1977, vol.1). These relations are the roots for the formation of "is a" hierarchies where the subordinates lie under their respective superordinates and inherit features for them. Such hierarchies are widely used in the construction of information-retrieval systems, where they are called inheritance systems (Touretzky, 1986). To show this relationship with an example, **pine** "is a" **tree**, therefore **pine** is the hyponym of **tree** and a **tree** is the hypernym of **pine**. Although it would be ideal to have a hierarchy in which every concept has one and only one hypernym, all semantic networks that exist today have occurences of many concepts which have more than one hypernym. In the Mikrokosmos hierarchy (Mahesh et al., 1995a), concepts on an average have five hypernyms. The causes and drawbacks of this multiple inheritance will be discussed in later chapters of this report.

Meronymy (commonly known as the Part-Whole or "has a" relation) is also a semantic relation between concepts. A concept represented by the synset {X,X',.....} is said to be a mernoym of the concept represented by the synset {Y, Y',.....}, if native

speakers of English accept sentences constructed from such frames as *A Y has an X as a part*, or *X is a part of Y*. For example **a tire** "is a part of" **a car** and **a fuse** "is a part of " **a electric appliance**.

Yet another relation that is similar to the meronymy is the holonym ("is a substance of") relation. A concept represented by the synset {Y,Y',.....} is said to be the holonym of the concept represented by the synset {X, X',.....}, if native speakers of English accept sentences constructed from such frames as *X is made from substance Y*, or *X is a substance of Y*. For example, **oxygen** is the holonym of **water** because **water** is made up of **oxygen**, or **oxygen** "is substance of" **water**.

The coordinate relation, on the other hand, is a semantic relation between concepts that occur at the same level in the hierarchy (i.e. have the same superordinate). A concept represented by the synset {X,X',.....} is said to be a coordinate of the concept represented by the synset {Y, Y',.....}, if and only if (X,X',.....} and {Y,Y',.....} have the same superordinate. Although not as commonly used as the other relations, this relation helps in computing similarities between concepts. For example, **Car** and **motor-cycle** are coordinates because both concepts have **motor-vehicle** as their parent.

Unlike the various semantic relations between concepts mentioned above, the antonym relation is a lexical relation between words. In general, two words are said to be antonyms if they express opposite concepts. Although in a formal sense the antonym of a word X is not X, strictly speaking this is not the case. We understand the known difficulty in defining the antonym relation through an example. The opposite of **rich** is **poor** because the two words represent opposite concepts but **not rich** is not the same as **poor**. In fact many people consider themselves neither rich nor poor. Although in general, we may assume that two words that represent opposite concepts are antonyms, this is not always the case. For example, **{rise, ascend}** and **{fall, descend}** may be conceptual opposites but the antonym for rise is not descend and the antonym for ascend is not fall and vice versa.

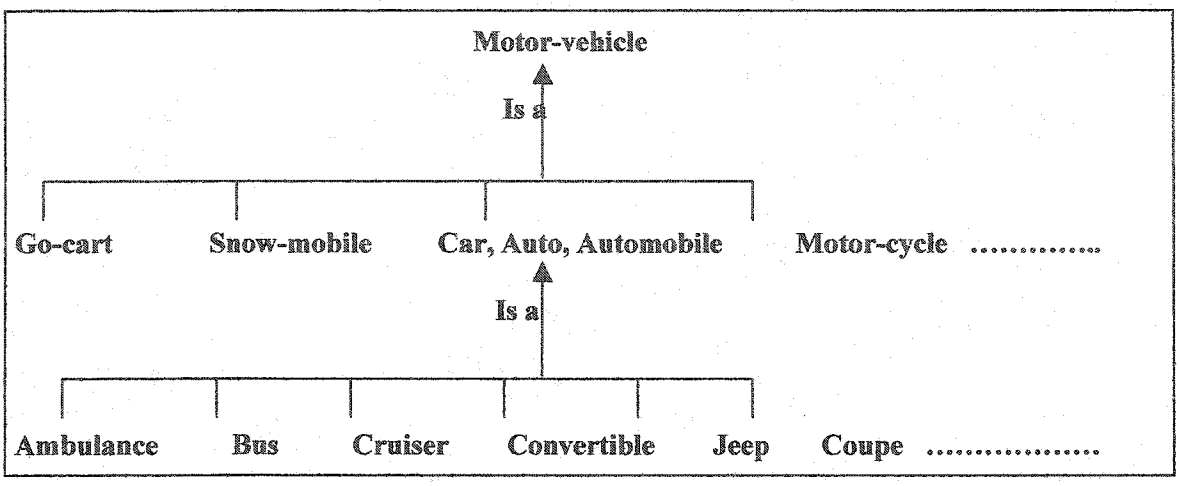


Fig 7: A conceptualization of the "is a" relation for the concept "car" in Wordnet.

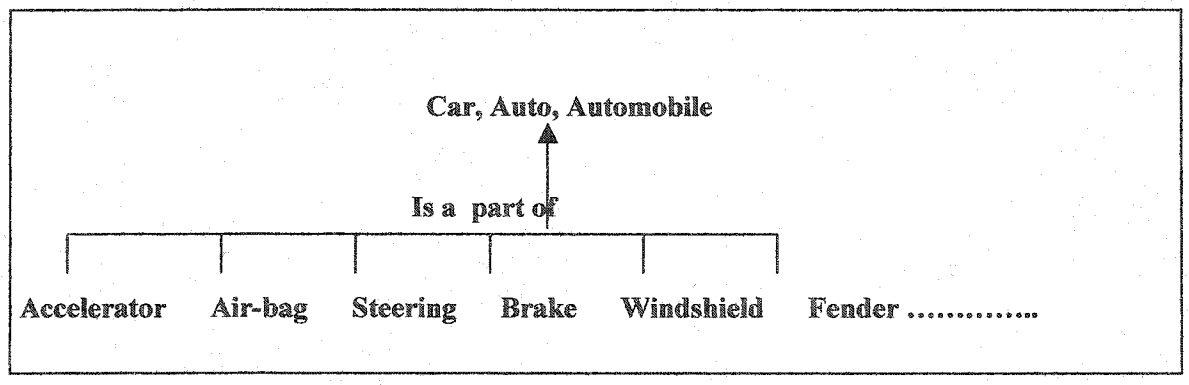


Fig 8: A conceptualization of the "part of" relation for the concept "car" in Wordnet.

Although many concepts in a semantic network need not have all these relations, the "is a" link is a necessity (i.e. the back-bone that drives the classification). Fig.7 & fig.8 illustrate one such meaning representation of a concept from the Wordnet semantic network.

As illustrated in fig.8, the concept **car** (as a motor-vehicle) can be represented by the synset **{car, auto, automobile}**, the set of hypernyms **{motor-vehicle}**, the set of hyponyms **{ambulance, bus, cruiser, convertible, jeep, coupe}**, the set of meronyms **{accelarator, air-bag, steering, brake, windshield, fender}**, the set of coordinates **{go-cart, snow-mobile, motor-cycle}** and a definition. The definition for this concept in

Wordnet reads "4 – wheeled motor vehicle; usually propelled by an internal combustion engine".

This overview of a lexicon, semantic networks, concepts & meaning representation, should help us define commonsense knowledge and understand its important role in NLU. Commonsense knowledge is the set of true facts we know about the world around us. Every one of us including children posses this form of factual information without knowing it. Some examples of such commonsense facts include :-

- A string can be used to pull an object, not push it.
- It isn't good to eat a string.
- You cannot make a balloon with a string.
- You can tie a balloon with a string.
- Birds fly.
- Snakes don't fly.
- People die whereas tables and chairs don't.
-

etc.

Lenat describes commonsense knowledge as the set of facts that are usually omitted in a conversation from fear of embarrassing the listener. Although commonsense facts are trivial enough that we do not need to explicitly state them in a communication, they play a very important role in helping us understand the message. It is a common belief in the knowledge engineering community that lack of such knowledge in computers is what restrains them from being able to understand sentences encoded in a natural language.

According to Minsky, the secret of what an entity X means to us lies in how our representations of X connects to the other things we know. If we understand something in only one way then we scarcely understand it at all because if something goes wrong with our only representation, we have nowhere to go. But if we had several representations of X, each integrated with its set of related pieces of knowledge, then when one of them fails we can switch to the other. We turn ideas and representations in our mind to

examine them from different perspectives until we find one that works for us, and that's what we mean by thinking (Minsky, 2000).

Commonsense knowledge can be further classified into what most semanticists call world knowledge and analytic knowledge. Relating these notions of commonsense knowledge to the perspective of the semantic network that we mentioned earlier, the nodes and the relations between them captured in the semantic network form the world knowledge component, whereas any inferences generated from the analysis of world knowledge fall under the analytic knowledge component. To enable NLU, we require a relevant contribution from both these components of commonsense knowledge. Of course it is important that the semantic network represents all the world knowledge that exists around us correctly before we can infer any form of analytic knowledge from the given world knowledge.

We briefly define the three levels of NLU primarily, syntax, semantics and pragmatics in an effort to relate them to meaning representation and commonsense knowledge. The word syntax comes from the Greek word *syntaxis,* meaning "setting out together or arrangement". The study of syntax deals with the structure of sentences in natural language, i.e. the arrangement of words in sentences (Jurafsky et al, 2000). Syntactic analysis (parsing) is the process of assigning a part-of-speech (noun, verb, adjective, adverb, ...) and grouping relevant words to form phrases. The parse is a simple application of production rules that represent the grammar of the language. Semantics on the other hand, deals with representing meanings and linking the meanings to lexical tags (words in the language). Semantic analysis deals with the analysis of meanings, one of the preliminary roles of which is the process of extracting the meaning for a given phrase. This is achieved by tracing the links from words to meanings and vice versa like those represented in the lexical matrix. Most times, to resolve ambiguities prevalent in the language, certain decisions have to be made to enable correct syntactic and semantic processing. These decisions are uncertainties that have to be resolved in order to understand utterances in the language. The study of reasoning involved to facilitate this form of decision making is known as pragmatics. In most cases, there are many

interpretations of a sentence that are both syntactically and semantically correct. It is the job of pragmatics to reason about the various interpretations and select the most plausible one. To illustrate the roles of syntax, semantics and pragmatics, let us try to understand the statement "3 / 0" and infer that this statement is invalid. The syntactic analysis is valid as it follows the production rule (Number ← Number Operator Number), the semantic analysis of the statement is "infinity", but it is the job of pragmatics to tell us that this meaning is plausible only in mathematics and is meaningless in other domains. Although in most practical applications it is difficult to strictly separate the roles syntax, semantics and pragmatics, we can in a broad sense ascertain that meaning representation and analysis is the job of semantics, and commonsense reasoning is the job of pragmatics.

3.5 Default Reasoning in Semantic Networks

Commonsense reasoning involves "jumping to conclusions". For example from the statement "the Toyota is parked on the street", we would conclude that it has four wheels. This is accepted as a truth by default because most Toyotas have four wheels. We use this probabilistic knowledge as a justification to this default rule. This form of reasoning is commonly known as default reasoning (Norvig et al., 1995). According to Shastri, any form of reasoning that permits us to infer properties of a given concept based on the properties of its ancestors, is reasoning in default. Default reasoning can be formally defined as the process of determining properties of a concept C, by looking up the properties locally attached to C, and if such local information is not available, by looking up properties attached to concepts that lie above C in the conceptual hierarchy (Shastri, 1989). Therefore in terms of Shastri's definition, "has part" 4 wheels is a property of the concept car, information that we acquire from the structure, and the ability to infer that Toyotas have 4 wheels because a Toyota is a car is reasoning with defaults. As simple as it may sound, many such inferences have to be made before we understand the sentence "the Toyota is parked on the street". Some other such inferences include truths like the Toyota has a steering wheel, it is a vehicle that can be driven, it is most likely that nobody is sitting in it, and so on. In most cases, one default truth may lead to another

thereby forming a chain. Although many researchers in the past have suggested that inferences can be chunked together and retrieved as necessary, it is impossible to group entire chains of inferences as there may be an infinite number of possibilities. Therefore some form of smart computation has to be performed to enable this kind of dynamic reasoning. Not only should the reasoning chains be dynamic, but they should also be computable at extremely high speeds because as a map to default reasoning in humans, we compute these chains so rapidly that we don't even realize the process involved in computing them.

In reality, further inspection might suggest that the Toyota parked in the street has only 3 wheels and is supported by a jack because the forth wheel is punctured. Although the default inference is not the reality in this case, we need to make such assumptions in-order to facilitate any reasoning about the car. It is better to know something that is valid 99% of the time than not knowing it all. These default rules hold as truths until further information is provided dictating a contradiction to what we already know. In an eventuality of new evidence that contradicts the default rule, all inferences justified by the default rule must be retracted. Retraction can be defined as the process of removing from the knowledge base previously made assumptions. We discuss retraction a little further to appreciate the seriousness of the problem, let P denote the inference that "Toyotas have 4 wheels" and Q denote the assumption that "Toyotas can be set in motion", before the contradiction in evidence, if we had inferred Q from P (i.e. $P \rightarrow Q$), then Q must also be retracted unless there is some other X that justifies the existence of Q. It is very important that retractions be performed accurately to avoid contradictions in the knowledge base, and thereby, in the inferences we make. This brings about the need for Truth Maintenance Systems (TMS) in knowledge bases. TMS's work towards either proving, disproving or finding an explanation to determine the validity of assumptions in the knowledge base with the aim of removing invalid assumptions. The validity of most assumptions can be checked as direct consequences of proofs from logic, but in some cases, there may not be enough evidence to determine the validity of the assumption, in which case we try to give a good explanation to justify the assumption. An example use of such assumptions may be to explain why a car does not start. There may not be enough

evidence to prove anything, but a good explanation is, "If we assume that there is gas in the car and that it is reaching the cylinders, then the observed absence of activity proves that the electrical system must be at fault." Technically the set of explanations E has to be composed of elements that entail P, which are either assumptions or statements that are true in the knowledge base. To achieve precision in explanations, E should be a minimal set, that is there is no proper subset of E that is also an explanation.

Most commonly known as exceptions, such contradictions along with techniques to rectify them (retraction) should be facilitated in the language used to represent knowledge. In other words, we need a logic that permits the non-monotonic growth of an initial set of beliefs with time and newer evidence. First order logic is strictly monotonic in the sense that it does not provide for the alteration or modification of previously made beliefs with time or newer evidence. Reasoning schemes such as default logic (Reiter 1980), non-monotonic logic (McDermott et al., 1980) and circumscription (McCarthy, 1980) are designed to handle such forms of retractions of previously made beliefs. Although these schemes have provided a framework for representing commonsense knowledge to facilitate default reasoning, all systems to date remain formally un-decidable and computationally slow.

A brief overview of how default rules are represented in default logic should shed light on how logic is extended to support non-monotonic reasoning. According to Reiter, a default theory (D,W) is defined comprising a set of first order sentences W and a set of default rules D (Reiter, 1980). Default rules are of the form :-

$$a : Mb/c$$

where **a** is the pre-requisite, **b** is the participant and **c** is the consequent. In essence this rule means that if **a** is true and **b** is consistent then **c** is true. In order to determine what is provable and what is not in a default theory, Reiter uses the notion of a fixed point. First of all, an operator G is defined on a set of first order sentences S as having the following properties :

1) W is a subset of G(S).
2) G(S) is closed under first order logic.
3) If a:Mb/c is a default rule in D and $a \in G(S)$ and $\neg b \in S$ then $c \in G(S)$.

E is then defined as an extension of the default theory (D,W) iff E is a fixed point of G i.e. G(E) = E. As a simple example to illustrate this :-

W = {bird(Tweety)}
D = {bird(x) : Mfly(x)/fly(x)}
E = {bird(Tweety), fly(Tweety)}

This logic helps us deal with exceptions like the case where Jacky is a bird that does not fly because it does not have a left wing, where the **Mfly(x)** which is the participant in the default rule does not hold. Such forms of exceptions occur in nature and should be accounted for in the representation to enable commonsense reasoning in default. The interpretation of the default rule can be read as if Tweety is a bird, and there is no exceptional evidence about Tweetys ability to fly then Tweety can fly.

Yet another form of exceptional cases are those of exceptional classes that occur within the inheritance hierarchy. For example, a penguin is a bird, and **fly(penguin)** is false, which contradicts the inherited property that **fly(bird)** is true. If we are given that Tweety is a penguin and although there is no exceptional information to contradict Tweety's ability to fly, we should still conclude that **fly(Tweety)** is false. These types of exceptions are captured by enhanced default rules of the form a:M(b $\wedge$ c) / b where c represents **~penguin(Tweety)**.

To describe the difference in exceptions we illustrate an example :-

W = {penguin(Tweety), bird(Penguin)}
D = {bird(x) : M(fly(x) $\wedge$ ~penguin(x)) / fly(x)}
E = {penguin(Tweety), ~fly(Tweety)}

Although this enhanced representation of the default rule captures exceptional classes like those of penguins and ostriches in this context, this poses yet another problem. How

do we infer that Tweety flies when all we know about Tweety is the fact that it is a bird. This ability to generalize is an ingenious ability humans posses. For Example, if we were asked the question, do most Russians drink vodka?, one would possibly assert that most Russians do drink vodka, not because we have factual information that 60% of the population of Russia drink vodka, but rather because of the fact that the only three or four Russians that we do know of do drink vodka. The use of probabilities, heuristics and smart algorithms on top of the knowledge structure to enable this form of a generalization is a suggested approach.

The current norm in classification of concepts in knowledge engineering assumes that it is perfectly natural for a concept to be organized in multiple hierarchies where a given concept may have more than one parent concept. This leads to the situation where properties inherited from both of the parents may be conflicting, specially so when the criteria for the classification are the properties themselves. For example, a concept **A** may have two parents **B** and **C** where most **B**'s may have the property **P** and most **C**'s may have the property **~P**. Then as defined by inheritance, the child inherits **P** and **~P** from both its parents, which leads to a contradiction. An example of such a situation is shown in fig 9:-

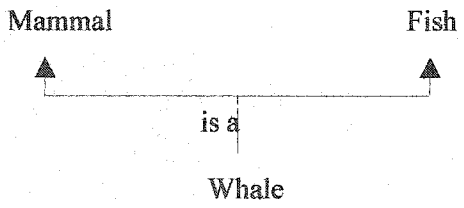


Fig 9: An example of multiple inheritance.

A whale is both a sub-type of a mammal and a fish. A whale therefore inherits the property of **has legs** from mammal and the property of **has no legs** from fish. Although many a proposition has been made to overcome problems that root from multiple inheritance, it still remains a challenge to restrain inference engines from encountering such contradictions. A proposed approach to overcome this problem is the use of and/or

inheritance, where the child inherits properties from selected parents based on their relevancy to the inference at hand. For a more detailed analysis on multiple inheritance and its problematic consequences we refer the reader to "Default Reasoning in Semantic Networks : A Formalization of Recognition and Inheritance", a paper by Lokendra Shastri in the Journal of Artificial Intelligence (Shastri, 1989). Although the knowledge engineering community is yet to see an ontology of world concepts that does not allow for multiple inheritance, this should not be overruled as a major hinderance to formalized reasoning with commonsense knowledge. In fact, current day reasoning systems have complicated the problem further more by treating problems that occur through multiple inheritance in a similar fashion as natural exceptions, i.e: treating inconsistencies in the structure we build in exactly the same fashion as we would treat inconsistencies in nature. A more detailed discussion about multiple inheritance in current day ontologies will be discussed in chapter 4, where we probe into a study of existent ontologies and their shortcomings.

3.6 Spreading Activation in Semantic Networks

The principle of spreading activation has been a traditional form of reasoning with semantic networks, neural networks and inference networks. As in the case of semantic networks, spreading activation (SA) dates back to the early work of some psychologists on associative memory in the mid sixties (Quillian, 1966). As discussed earlier in this survey, a semantic network can be visualized as a graph where concepts are the nodes and relations between concepts are edges that connect these nodes. The notion of a SA is that of exchanging signals along edges of the graph. All nodes are initially taken to be idle and are activated by signals from either the input or other neighboring nodes. Two nodes are said to be adjacent to each other if there is an edge connecting them. We further define spread as the transmission of the signal from one node to another. One outcome of the spread is the path; i.e. a series of connected nodes. In general, a path corresponds to an inference. Specific techniques used to spread the activation differ from one problem to another, and more specifically from one application to another. The more commonly used

approaches in spreading activation are the pure spreading model, the constrained spreading model and the spreading activation with feedback.

3.6.1 The Pure Spreading Model

The pure spreading model is the simplest method to spread activation across the network. This technique triggers multiple iterations of sequential spreading until halted either by a terminating condition, or the user. The iteration is composed of one or more pulses and a termination check. A pulse can be broken down into pre-adjustment, spreading and post-adjustment phases. The post-adjustment and the pre-adjustment phases primarily share the same goal in all models of SA where their functionality is to create a loss of interest, commonly known as decay of active nodes. This is a technique used to terminate the activation of an iteration to enable future iterations to activate the node. This interest to capture multiple activations of the node is application dependent and therefore may not be found in all implementations of the SA. In fact this argument builds to explaining why both the pre-adjustment and post-adjustment phases are only optional and therefore not necessarily used in all models of the spreading activation.

The spreading phase, on the other hand, consists of a network of transmissions between nodes. There are many possible ways to spread an activation across the network. The criteria for distinction between various models of spreading activation roots from the methodology used in spreading. To better understand the spread, all links in the semantic network have pre-assigned weights that determine the strength of the link. Initially some arbitrary value is chosen as a measure of the input, and this value is then fired to relevant nodes in the network. Each node activated in the iteration computes its output value which is a function of the input value and the strength of the link through which the activation had to pass through to access the node. After the node has computed its output value, it in turn fires to all its adjacent nodes, usually sending the same value to all of them. The activation spreads in this recursive fashion to reach nodes that are far off from the root (nodes from where the paths are initiated) of the spread. After a number of such pulses have been fired, the termination condition is checked. If the condition is satisfied,

the process is stopped, otherwise another series of pulses is fired, thereby making the firing of pulses in itself an iterative process. The final result of spreading activation is the activation level of the nodes at the time of termination.

Although very simple to implement, the possibility of flooding the entire network and the inability of this method to use semantic information present in the links are some of its limitations. Although the proper use of pre and post-adjustment phases can provide reasonable solutions to avoid flooding in the network, various constraints used in constrained spreading activation prove better (Crestani, 1997).

3.6.2 Constrained Spreading Activation

Constrained spreading activation, more commonly known as priming is a technique that avoids flooding of the network by the use of constraints at every node to restrict the spreading across edges that represent relations relevant to the input set. This can be achieved by implementing a set of rules at every node to control the spread. Constrained spreading activation not only avoids clogging the network, but also makes use of semantic information prevalent in the links. Some constraints used in this model include:-

Distance constraint: This requires the nodes in the semantic network to be ordered according to their semantic distance. The relations between adjacent nodes are first-order relations. Relations with one intermediate node are second order relations, relations with two intermediate nodes are third order relations and so on. Depending on the application in concern, the distance can be limited to first, second or third order relations.

Fan-out constraint: The fan-out constraint involves the termination of the spread at nodes that have a large number of adjacent nodes. This approach not only reduces the possibility of clogging the network, but also restricts the activation to a narrow spread thereby avoiding derivations from nodes that represent broad meanings. Although a fan-

out constraint provides for narrow spreading, possible loss of important inferences occurring from nodes with broad meanings should be anticipated.

Path-constraint: In this form of a constraint, the spread is restricted to preferred paths that branch out of a node. This is a decision based on the semantic relatedness of the path, modeled using labels and weights on the links. This approach stops the unnecessary activation of meaningless paths.

Activation constraint: The activation of a node during the spread can be enhanced to compute its output based an a threshold function, a measure of the entire activation prevalent in the network at the given time.

3.6.3 Spreading Activation with Feedback

Another important enhancement to spreading is by the modification of existing constraints based on feedback about the activation levels between pulses. This evaluation and modification is an external process usually performed by the user. If the application demands only a trivial evaluation and modification phase, which can be automated, an external process can be employed to fulfill this task. One important application of this model is the retrieval of information based on user feedback. This feedback can be used either in pre-adjustment phase to further direct the spread or in the post-adjustment phase to modify the existing path of the spread.

According to Corriveau, there is a perfect map between the processing model supported by the Object Oriented Paradigm (OOP) and that required by spreading activation (Corriveau, 1994). He argues that the concurrent processing and parallelism required in SA is well supported by the OOP. Owing to the fact that mechanisms underlying SA at the high level have remained unchanged since their introduction to AI in the sixties, as emphasized by Quillian and Fahlman, Corriveau implemented a generic framework for SA using Smalltalk and ABCL/1. This framework supports the modification of specifics of the SA based on the demands of the application.

3.6.4 Marker-Passing Approaches as a reasoning model for NLU

The marker passing approach to a spreading activation can be broadly classified as an enhanced spreading activation with constraints. This model assumes a traditional semantic network where concepts are represented as nodes and the associations between concepts are represented as links. The communication between nodes is accomplished through messages called markers moving through the network according to propagation rules. Under this model, inferences are derived based on set intersections and transitive closures. The set intersection locates all nodes that share common properties whereas the transitive closure handles inherent relations in the semantic network like inheritance and meronymy. Two basic forms of reasoning used in a semantic network are default reasoning and reduction (Shastri, 1989). As a direct map, set intersection can be used to facilitate reduction and transitive closure operations can be used to handle default reasoning. Default reasoning and reduction are described at greater length in the next section.

The beauty of the marker-passing model lies in its ability to incorporate functionality into the marker that enables the marker to modify its path as a consequence of the nodes that fall in its path. This is achieved by dynamically computing propagation rules as consequences of interactions between the marker and the nodes that lie in its path. In addition to the initial propagation rules determined by constraints implemented as links in the semantic network, the functionality of the marker can be used as an added modifier. Some commonly used constraints implemented into the processing of the marker include time decay, accumulation of relational strength and elimination of irrelevant inferences. Time decay is a measure of how many times the marker has been called to pass marks (nodes) from a new origin. Time decay is an important technique to prevent flooding of the network by elimination of a marker that is unable to perform its task during its life time. The life time of a marker is pre-defined based on the density of the network, permissible marker traffic and the intended form of reasoning. The time decay used by WIMP, a program built to understand English is exponential (Charnaik, 1986). The accumulation of strength on the other hand is a measure of marker relevancy computed

from all nodes visited in its path. This deems rather useful in elimination of paths to select more plausible paths (inferences) and to further rank the selected ones. The ability of the marker to eliminate certain nodes along its path can be embedded into the functionality of the marker. This is usually based on heuristics and statistical analysis of test results. Examples of such heuristics embedded in SNAP, a program that extracts information from real messages include deeming shorter paths as more relevant inferences than longer ones (Maldovan et al., 1996). SNAP filters out irrelevant paths and ranks good ones by propagating syntactic and case constraints along with the markers. Whenever these constraints are not satisfied, the respective propogation ceases. Any colliding marker or trajectory indicates a valid semantic path (inference) in the knowledge base, and when two or more markers are passed to the same node, a marker collision occurs. Without any loss of generality we can state that, one half of the generated path originates from the initial representation whereas the other half leads to the inferred node. Each collision denotes a possible inference. Therefore the final process is to evaluate the paths and determine if they are selected, rejected or deffered. Most applications of the marker-passing model implement a separate module called the path-checker which accomplishes this task based on fixed, pre-determined a priori criteria. An example of such a trajectory or collision is illustrated in fig.1.

As discussed earlier, the suggested approach to NLU is through the use of a large knowledge base that provides commonsense knowledge deemed important to comprehend natural language. Therefore the process of reasoning with the knowledge base can be mapped to a search, where the search key is the sentence and the result of the search is the intended interpretation of the sentence. The search agents in this context are the markers (data structures) that travel through the knowledge base represented as a semantic network, based on propagation rules to identify intended interpretations. Our beliefs about NLU are supportive of Charnaiks claim that "a standard platitude is that understanding something is relating it to what one already knows. The exact nature of such relating is not obvious, but one extreme example would be to prove that what one is told must be true on the basis of what one already knows" (Charnaik, 1986). In essence, we are under the notion that reasoning about a concept can be accomplished based on the

relatedness of the concept to other concepts in the knowledge base. Previous implementations of such models to reasoning with semantic networks have been used to attack problems in NLU/NLP. Some efforts in this direction include SCISOR, a question answering system (Rau, 1987), SNAP, a system built to extract information from real text messages (Moldovan et al, 1996) and WIMP, a wholy integrated marker parser (Charniak, 1986).

3.7 Metaphors, Metonymy and Conventional Schemas

In our day-to-day use of language, we often use words and phrases to express concepts that are totally different from those that are appropriate to the lexical structures. For example the sentence "John kicked the bucket", is used to convey the meaning "John died". In this example there is no correlation what so ever between the literal meanings of "kick", "bucket" and "death". Even so we seem to have no problem in using and understanding such statements in language. This notion of using sentences to represent meanings other than their literal meanings is known as metaphor. Although a very intriguing and fascinating feature of natural language, metaphors are nightmares in NLU. Simply so because no conventional approach to semantic analysis has an incorporated facility to decipher intended meanings of metaphors. It is a commonly accepted notion that metaphors are largely responsible for any polysymy present in natural language. Although the example we listed above is a very harsh one and one may well claim that such sentences fall within the horizon of idioms, metaphors can occur more subtly in language. In the sentence "if Ford changes its mind, it will buy the BMW Corporation", Ford is an organization that does have a mind in the literal sense, but in this case is a representative of the minds of the people in the management. Similarly in the phrase "a pool of information", although one cannot put or remove information in a pool, we use pool to denote a collection. More important than the harsh metaphors, such forms of subtle metaphors pose an immediate threat to understanding language due to the frequency of their use in everyday communication.

It still remains an open question as to how we understand the intended meanings, although the words and phrases used to represent them have nothing to do with them. Understanding metaphors has been a prominent area of study within NLP/NLU and linguistics alike, and many a pioneer has tried to explain their use in language. The most widely accepted attempt is that of kinesthetic schemas where it is believed that many, if not most, of the metaphorical expressions that we encounter every day are motivated by a relatively small number of so called conventional schemas (Lakoff et al., 1980). Conventional schemas are conceptual templates we form about concepts that we know of and experience at the basic level. The basic level is the level at which we as humans interact with the external environment, characterized by gestalt perception, mental imagery and motor movements. At this level people function most efficiently and successfully in dealing with dis-continuities in the natural environment. It is at this level of reasoning that we distinguish tigers from elephants, chairs from tables, roses from daffodils and asparagus from broccoli (Lakoff, 1990). These conceptual templates are then mapped to concepts from other levels. An example of one such schema is the container schema (Johnson, 1987) which consists of a boundary distinguishing an interior from an exterior . The Container schema defines the most basic distinction between in and out. We understand our bodies as containers and the most basic things that we do are ingest and excrete, take air into our lungs and breathe it out. According to Lakoff and Johnson alike, this notion of **in** and **out** is then mapped to concepts at other levels. Example uses of **in** and **out** in a very commonly used conversation are listed below to demonstrate this conceptual map.

1) I woke **out** of a deep sleep and peered **out** from beneath the covers in my room.
2) I pulled my self **out** of bed and climbed **into** my robe.
3) I stepped **out** of the bedroom and **into** the bathroom.
4) I looked **into** the mirror and saw my face staring at me.
5) I reached **into** the cabinet and took **out** the toothpaste.
6) I squeezed **out** some tooth paste onto my toothbrush and put the tooth-brush **into** my mouth.
7) I brushed my teeth and rinsed **out** my mouth.

Such in-out moves are very common in our daily lives and this relationship can be accounted for by defining conceptual links between concepts from different levels that abide with the container schema. Although many schemas like the Container schema provide reasonable explanations to the use of metaphors, there exist many unanswered questions. Is this how we truly use lexemes metaphorically? How many such conceptual schemas are there? How do we account for this map? How will understanding these maps help in understanding language? Some other such kinesthetic schemas suggested by George Lakoff in the book Women, Fire and Dangerous Things include :-

1) The Part-Whole schema.
2) The Link-Schema.
3) The Center-Periphery schema.
4) The Source-Path goal schema.
5) Up-Down schema.
6) Front-Back shema.
7) Linear-Order schema.

Intriguing as they are, metaphors have posed important open problems to researchers in NLU. The complexity in their analysis can be illustrated by a simple question. How can a word or phrase be used to convey a meaning that it does not stand for and yet be understood? Although so far we have generalized all problems of this nature to those of metaphors, an important subclass of problems are those caused by metonyms. Metonyms are situations where concepts are denoted by the name of other concepts that it is closely related to. According to Lakoff, metonymy is one of the basic characteristics of cognition. The notion of taking one well-understood or easy-to-perceive aspect of something and using it to stand either for the thing as a whole or for some other aspect of it is a norm in human conceptualization. Sentences (8) to (10) illustrate the use of metonyms:-

8) The ham sandwich paid for the table.
9) Kuwait has become a Nagasaki.
10) University of Windsor has announced the date of the graduation ceremony.

Here in the literal sense these sentences make no sense because a ham sandwich cannot pay a bill, Kuwait cannot become a Nagasaki and the University of Windsor in itself cannot announce anything. But as metonyms of a person who ate a ham sandwich, the war in Kuwait and the management of the university, these sentences make perfect sense. When in a common conversation, we are presented with such metonyms, we never seem to ask responsive questions like how can a ham sandwich pay a bill?, how can a dumb university announce anything?, and so on, but instead we seem to comprehend the intended representation with minimal effort. This leaves us room to believe that not only is there some structure that restricts the use of metaphors and metonyms, but this structure should enable this form of reasoning with minimal effort (i.e. by default). Although not drastically different from metaphors, we discuss them in isolation because metonyms dictate a very important feature of cognitive reasoning. The ability to represent a category of concepts by either a prototype concept or one of its attributes, is generalization in a very broad sense. This ability should not be trivialized, not because many a researcher in AI believes that the ability to reason is synonomous to the ability to generalize, but because of the ease of representation that metonyms and metaphors provide us in representing concepts and in many cases entire classes of concepts. Although there is no formal proof, it is not unreasonable to assume that concepts in the network that do not have any synsets are represented by metaphors/metonyms. A technique that facilitates the representation of concepts, regardless of their names (synonyms).

3.8 Thematic Roles And Selectional Restriction

As important is the map between lexemes to meanings, so important is the map between meanings to lexemes. This bi-directional relationship shared by these entities is a complicated intricate detail that is a challenge for the best of linguists to understand regardless of the fact that it is the most trivial of applications humans use in every day conversations. In any form of NLU, it becomes impossible to ignore that any algebra of meanings performed at the semantic (conceptual) level can be represented by an algebra

of lexemes at the syntactic level. Therefore an analysis of rules and constraints used in the generation of a composite of lexemes should reveal the rules we use to break down these composites, and should thereby enable a composite of meanings to be broken down. In this set of constraints, not only do we refer to syntactic but also semantic and pragmatic constraints. The use of selectional restrictions in NLU/NLP is an implementation of this notion. If we view primitive lexemes (nouns, verbs and adjectives) as predicates with arguments, selectional restriction involves the process of constraining the domain of each of these arguments to a subset of the lexicon. It is not abnormal that more than one argument of the same predicate is restricted to the same subset. These form of selectional constraints not only provide us with important cues to decipher a lexeme but also help optimize the lookup for an argument by narrowing the search space.

To appreciate the use of selectional restriction in NLU, we make an effort to understand thematic roles and some representations used to implement these constraints. Thematic roles are a set of categories that provide a shallow semantic language for characterizing certain arguments for verbs (Gruber, 1965), (Fillmore, 1968). Although Gruber and Fillmore concentrate more towards thematic roles for verbs, it is important to note that adjectives and most nouns also have categories of arguments too. Sentences below illustrate thematic roles for an example from each of these parts-of-speech.

1) Eats(x)
2) heavy(x)
3) Fare(x)

In a very general explanation of a thematic role we consider sentences (1) to (3). Without the loss of generality the choice of predicate entities was restricted to less ambiguous lexemes to simplify the explaination. In sentence (1) we define the thematic role of the argument **x** in the intransitive verb **eats** as an agent whose type can be restricted to **animal** and every concept that is a sub-type of **animal**, primarily so because only animals eat. Similarly in sentence (2), the thematic role for the argument **x** in the adjective **heavy** can be defined as a subject whose type can be restricted to **physical entity** and all its

hyponyms. In sentence (3), the thematic role played by **x** in the noun **fare** is that of a modifier and can be restricted to nouns falling strictly below the concept **modes-of-transport** like **train**, **bus**, **air** and so on. Intransitive two-place verbs, on the other hand, require a more detailed analysis in determining thematic roles, primarily because the second argument can be restricted to multiple roles.

4) John ate an apple.
5) John eats in a restaurant.
6) eat(x,y)

The template of the predicate **eat** for both the sentences remains the same as shown in sentence (6), where the argument **y** could either be **an apple** which is **a fruit** or **a restaurant** which is **a location**. As a suggested solution to determining such templates for transitive verbs, we propose the identification of combinations of prepositions that could possibly accompany the verb. Regardless of how the breakdown of templates is derived, it is important to appreciate that not only do these thematic roles help in syntactic, semantic and pragmatic interpretations of the sentences, but they also determine important criteria for classification. Although this strategy towards the classification of concepts has never been attempted in the past, it cannot be overruled as an elegant approach to using lexical constraints to classify semantic concepts. The benefits of the use of thematic roles in NLU has led many a researcher to a detailed study in this domain. The lack of a standard set of thematic roles and the existence of ambiguities as a hindarance to identifying these roles, some prototypical patterns and heuristics have been suggested to deal with these inconsistencies (Fillmore, 1968), (Jackendoff, 1972). Fig.10 lists thematic roles used by computational systems, along with a definition to describe each role and fig.11 illustrates an example of the use of each role in the language (Jurafsky et al., 2000). In fig.11, the italicized parts of the example sentences denote the occupants of the various thematic roles.

Selectional restrictions can be used to augment thematic roles by allowing lexemes to place certain semantic restrictions on lexemes and phrases that accompany them in a sentence. More specifically, a selectional constraint is a semantic constraint imposed by

the lexeme on the concepts that can fill the various argument roles associated with it (Jurafsky, 2000). The violation of a selectional restriction is easily noticeable and under the assumption that all possible thematic roles that can occupy an argument in a predicate have been accounted for, we may treat such violations as malformed sentences. The discussions so far have ignored ambiguous lexemes. Let us consider examples of ambiguous lexemes to see how these restrictions help in lexical disambiguation. In sentences (7) and (8), the verb **serve** is the ambiguous lexeme.

7) Air-Canada flights always serve dinner for over night flights.
8) Air-Canada flights serve Denver.

Thematic Role	Defintion
Agent	The volitional causer of an event.
Experiencer	The experiencer of an event.
Force	The non-volitional causer of an event.
Theme	The participant most directly affected by the event.
Result	The end product of an event.
Content	The proposition or content of a propositional event.
Instrument	An instrument used in an event.
Beneficiary	The beneficiary of an event.
Source	The origin of an object of a transfer event.
Goal	The destination of an object of a transfer event.

Fig 10: Some commonly used thematic roles and their definitions.

Thematic Role	Example
Agent	***The waiter*** spilled the soup.
Experiencer	***John*** has a headache.
Force	***The wind*** blows debris from the mall into our yard.
Theme	Only after Benjamin Franklin broke ***the ice*** .....
Result	The French government has built a ***regulation-size base-ball diamond***.....
Content	Mona asked "***you met Mary Ann at a supermarket***"?
Instrument	He turned to poaching catfish, stunning them with a ***shocking-device***....
Beneficiary	Whenever Ann Callahan makes hotel reservations ***for her boss***.....
Source	I flew in ***from Boston***.
Goal	I drove ***to Portland***.

Fig 11: Prototypical examples of various thematic roles.

In sentence (7) we refer to the cooking sense of serve, where as in sentence (8) we talk about the sense that denotes a commercial service. By identifying the thematic role of the argument, we can identify the right meaning of serve. In sentence (7) serve will successfully be identified as the cooking sense of serve as it accepts as an argument a concept of type meal, and similarly the commercial service sense of serve is restricted to some form of a geographic entity. A similar approach in reverse can be applied to identify the correct senses of ambiguous arguments. In the context of anaphora resolution, let us consider sentences (9) and (10).

9) The dog chased the cat.
10) It barked as it ran.

In sentence (10) there are two references which have to be solved. The use of selectional restriction helps solve the first, because we know that dog's bark therefore the first **it** in sentence (10) must refer to the dog. But in the case of the second occurrence of the pronoun **it**, these restrictions do not tell us much because both cats and dogs run. Under such an exceptional circumstance which is not very common, we use some heuristics to determine the more possible reference. One such heuristic applied here is: if there is no violation of constraints and the two pronouns are the same, they refer to the same object. Let us consider yet another set of sentences (11) and (12) to see if this heuristic still holds.

11) The dog chased the cat.
12) It ran after it for at least 2 kms.

In this case the first occurrence of **it** refers to the dog where as the second occurrence refers to the cat. But assuming the constraint run_after(x,y) → x != y in the commonsense knowledge base, the second reference will be correctly solved. To better understand how selectional constraints are represented at lower levels, we consider the sentence **John ate a hamburger**. When such a sentence is analyzed, the semantic contribution of the verb **eat** looks like the following:-

$\exists$e,x,y Eating(e) $\wedge$ Agent(e,x) $\wedge$ Theme(e,y)

This representation simply states that y, the filler of the thematic role, is associated with the eating event via the theme relation. So far there is no constraint on what y should be. To incorporate the constraint that y should be something edible, we simply conjunct the constraint to the representation.

$\exists$e,x,y Eating(e) $\wedge$ Agent(e,x) $\wedge$ Theme(e,y) $\wedge$ Isa(y, edibleThing)

To further add the restriction that the agent of the **Eating** event should be a subtype of animal, we conjunct yet another constraint to the representation.

$\exists$e,x,y Eating(e) $\wedge$ Agent(e,x) $\wedge$ Theme(e,y) $\wedge$ Isa(y, edibleThing) $\wedge$ Isa(x, Animal)

So when the sentence "John ate a hamburger" is encountered , the semantic analyzer forms the following representation:-

$\exists$e,x,y Eating(e) $\wedge$ Agent(e, x) $\wedge$ Theme(e, y)
$\wedge$ Isa(hamburger, edibleThing) $\wedge$ Isa(John, Animal)

Information from the commonsense knowledge base reveals that a **hamburger** is an **edibleThing** and **john** can be the name of an instance of **animal**. Since in this example all the constraints of the representation are satisfied, the sentence can be treated as well-formed. On the other hand, if either of the predicates were not satisfied, the entire representation would be false and thereby lead to a selectional violation.

Chapter 4

Overview of Existing Ontologies

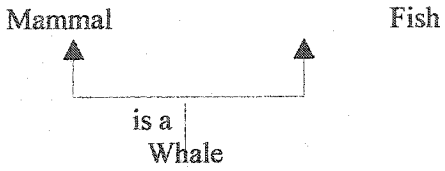
Having distinguished the boundaries of commonsense and domain specific knowledge, we visit some of the commonly known ontologies and semantic networks. Keeping in mind the focus of this survey, we exclude domain specific knowledge bases from the list. Having felt the need for commonsense reasoning in language understanding, the AI community has been waiting with open arms since the early eighties for a unique structure of commonsense concepts that enables commonsense reasoning. The enormous work hours required to build such a structure has been a hurdle to many efforts in this direction. As a consequence, there have been only very limited efforts, most of which are not yet complete. The five most commonly known commonsense-like ontologies in the NLU community are the Cyc knowledge base, the Wordnet semantic network, the Mikrokosmos ontology, the Generalized Pennman Upper Model and the EDR electronic dictionary. Although we can in a certain sense identify these efforts as knowledge bases for commonsense reasoning, each of these structures lie somewhere between a conceptual

dictionary and a commonsense knowledge base. To further distinguish them, Wordnet, EDR and Mikrokosmos are more of conceptual networks where as Cyc is more towards a commonsense knowledge base. The end goals of these ventures revolving around the same domain, they share many a common feature, whereas the specific motivation behind each of these projects being different, they differ on many issues. The most basic feature shared by all the above listed conceptual structures is the definition of the "is a" relationship between concepts based on common properties shared by all elements that belong to the classification. Although it is a well established notion that the inheritance relationship between concepts is a stepping stone to generalization, the techniques used to derive this relationship may however differ. Defining inheritance based on common properties is one such well-accepted methodology.

The success enjoyed by these projects in completing their respective ontologies leaves no room to doubt the importance of their approaches followed to defining the inheritance relationship necessary for the classification. But as a consequence, all these structures share a common flaw, the occurrence of multiple inheritance. The notion that a concept may inherit the properties from more than one parent is the primary root to this problem. Although at the conceptual level this may seem trivial and tempting, it is very crucial in the reasoning that follows. Primarily because if multiple inheritance is wrong, then the entire ontological commitments entailed by the structure are wrong, and thereby any inferences made using this knowledge base are prone to error. More technically, the use of multiple inheritance in these structures leads to contradictions between axioms in the knowledge base. A major portion of the seriousness of mulitple inheritance roots from the fact that most commonsense reasoning is default (i.e. based on the inheritance relationship). As listed earlier in this survey default reasoning and reduction are the quintessence of commonsense reasoning (Shastri, 1993). Fig.12 illustrates one such possible contradiction that occurs due to multiple inheritance.

It is important to recognize that one axiom leads to the derivation of another and any wrong inference along this chain of derivation leads to funny results. A state in point, most knowledge bases that allow for multiple inheritance have some form of a

contradiction re-solver that crawls the knowledge base to identify and rectify contradicting axioms (inferences). An instance of such a backend process is the contradiction resolver used by Cyc.



E1 : Whale(x) → Mammal(x)	(committed by the structure)
E2 : Whale(x) → Fish(x)	(committed by the structure)
E3 : Mammal(x) → ~ Fish(x)	(committed by the structure)
E4 : Fish(x) → ~ Mammal(x)	(committed by the structure)
E5 : Whale(x) → ~ Fish(x)	(E1 & E3, transitivity of →)
E6 : Whale(x) → Fish(x) ∧ Whale(x) → ~ Fish(x)	(E2 & E5, P,Q → (P∧Q))
Therefore a contradiction.	(E6, P→Q ∧ P→~Q)

Fig 12: Contradictive axioms due to multiple inheritance.

Although many solutions like Shastri's entropy and McCarthy's extension of first order predicate logic to handle exceptions have been suggested in the past, these implementations have not been able to eradicate completely the ill effects of multiple inheritance. Knowing that multiple inheritance is a sin in building ontologies for commonsense reasoning, we are unaware of any current effort to build an ontology that refrains from it.

To understand some differences in the approaches used by CYC, Wordnet and ERD, we analyse a debate between Doug Lenat (the founder of CYCorp), George Miller(the founder of Wordnet) and Toshio Yokoi (the founder of EDR) in the Critiques and Responses published in the communications of the ACM, 1995. Doug Lenat states that differences between his and Miller's approaches stem from four sources which include :-

1) The precision in defining terms and methodology.
2) The importance of words in natural languages like English.
3) A small number of key semantic relationships.
4) Disambiguating word senses to distinguish the linguistic contexts in which the words are used.

In significance to definitions from Wordnet, Lenat argues there might be many hidden assumptions that were overlooked when defining concepts. He further suggests that the methodology used to codify and represent knowledge is incomplete. Moreover, according to Lenat words in English are just a result of historic accident rather a grand plan. By this he means that there is a need for much more than lexical information to define concepts with respect to which he bases the next argument that Wordnet has only six types of relationships that link concepts, whereas Cyc has thousands. Lastly there is a difference in the approaches suggested by Lenat and Miller to perform NLU/NLP tasks like disambiguation, which is of course a consequence of their respective structures. Most of these differences apply not only to Wordnet but also to EDR. On a slightly different note, Lenat also comments that both Wordnet and EDR have given significantly less importance to defining concepts represented by multiple synsets like "bachelor party", "JFK assassination", and so on.

Miller, on the other hand, supports Lenats claim that there is a need for something of a greater amplitude than Wordnet to enable commonsense reasoning. He illustrates some of the difficulties one may encounter in hand-crafting such a massive knowledge base. Miller also states that both EDR and Wordnet assume that linguistic knowledge can be separated from commonsense knowledge. He further adds that lexical knowledge (knowledge of words and their meanings) is necessary for computers to process human languages, and that a large but finite number of factual propositions (definitions and linguistic relations) can satisfy this need. Yokoi agrees to most of the comments made by Lenat and Miller, apart from which he emphasizes the notion that language is the key to organizing general knowledge or world sense. Lenat's response to Yokoi's statement was contradictory primarily because Cyc's belief to organizing commonsense knowledge does not assume language to be the key. To defend his argument, Lenat states that birds

existed way before we learnt to talk about them, so concepts exist regardless of the words we use to represent them, and should therefore be organized regardless of the lexicon. He further justifies himself by adding that deaf and dumb people are no more stupid than anyone else. Although we do not argue against the claim that concepts exist regardless of the language used to define them, we support Yokoi's claim that language can be used as the key to organizing existent concepts. Primarily so because of the map between words in the language and the existent concepts. In fact, what better tool do we have apart from language itself to understand all that language can represent. Although language might have been an accident rather a grand plan, it is the most commonly used media of human interaction. In fact, it is the only interface to communicate what we know.

4.1 CYC : A Knowledge base for Commonsense

Cyc is a bold attempt to assemble a massive knowledge base [on the order of 10^8 axioms] spanning human consensus knowledge (Lenat et al, 1990). This ambitious long-term high-risk commercial venture has been alive since 1984. The motivation behind this effort is to build a structure that enables intelligent reasoning by software. As the business week magazine describes this cyber thinker of tomorrow, Cyc is an ambitious version of the old-school, top-down system. A humongus effort that has swallowed some 40 million dollars and 15 years of time, trying to organize its reasoning engines and stuffing its knowledge base with half a million rules derived from over 2 million commonsense facts. The kinds of things that we learn during child-hood: Mothers are always older than daughters, birds have feathers, people remain dead once they die, a ball thrown in the air will fall down, and so on. Cyc was first started in 1984 at Microelectronics & Computer Technology Corp. (MCC), an Austin-based research consortium of high-tech companies. For ten years MCC was responsible for Cyc's growth. A crew of linguists, philospphers, anthropologists and engineers spoon-fed data into Cyc. In 1994, MCC became CYCorp and has been making profits of about 3 million dollars each year ever since. Although Cyc is less than half way complete, limited-purpose versions of Cyc are being used by companies like Glaxo Welcome, Digital

Equipment, IBM and United health care. Glaxo Welcome and United healthcare use Cyc to manage huge online thesauruses of pharmaceutical and health care terms. Lenat hopes to have a complete Cyc by the year 2020.

As per the stance of Cyc to date, only a small part of this knowledge base of approximately 3000 terms has been made public. It is believed that this portion contains the most general concepts of the human census reality. Cyc in its entirety contains over 14000 English root words in its lexicon with word class and subcategorization information plus their mappings into the KB. Each concept in the knowledge base is represented as a Cyc constant, also known as a term or unit. Each term has "is a" links to superclasses of which it is an instance, plus GENLS links to superclasses of which it is a subclass. Two most important Cyc classes are collections and relations (predicates and functions). Apart from the "is a" and GENLS links, collections also have IsASubsetOf links. Illustrated below is a textual representation of two sample Cyc constants – a collection and a relation. Each has a heading, an English gloss and one or more relation attributes indicating links to other constants in the knowledge base.

The collection of all heads of #$Animals.

Is a :
#$AnimalBodyPartType #$UniqueAnatomicalPartType

Genls :
#$AnimalBodyPart #$BiologicalLivingObject

Some Subsets :
#$Head-Vertebrate

#$hairColor
<#$Animal><#$ExistingObjectType><#$Color>
(#$hairColor ANIMALBODYPARTTYPE COLOR) means that the hair which the #$Animal ANIMAL
has on its BODYPARTTYPE has the #$COLOR. For example (#$hairColor #$SantaClaus #$Chin
#$WhiteColor). This is normally #$Mammal hair, but certain #$Invertebrates also have hair.

Is a :
#$TernaryPredicate #$TangibleObjectPredicate

arg2Genl :
#$AnimalBodyPart

Although Cyc has received great attention by media and industry alike, many researchers believe that the structure underlying this knowledge base is ad-hoc. Cyc addresses the problems of a large scale knowledge representation of commonsense world by using an extremely ad-hoc approach based on heuristics and introspection (Guarino et al., 2000). Cyc being way ahead in this race to commonsense reasoning, has been applauded by may a cognitive scientist for taking these risks. But we must not ignore the illness of problems caused by multiple inheritance because any conflicting information available cannot be resolved by the location of the concepts in the conceptual hierarchy, primarily because this is the root of the problem. Unlike multiple inheritance, other sources of conflicts like class exceptions and instance exceptions can be treated at the conceptual level (i.e. by the location of concepts in the hierarchy). An illustration of multiple inheritance in the Cyc taxonomy is shown below :-

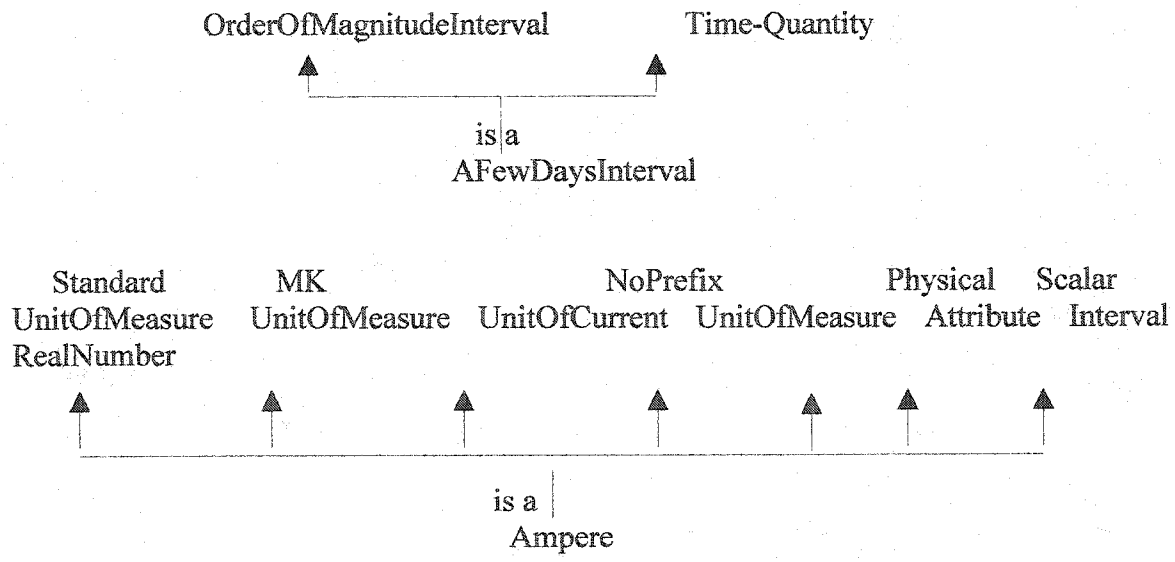


Fig 13: Instances of multiple inheritance in the Cyc knwledge base.

4.2 The Wordnet Semantic Network

Unlike the motivation behind Cyc, which is a commonsense reality, Wordnet is an effort to create a semantic network of lexical relations. This semantic database is a composite of three primary constituents. The first and foremost is a taxonomy for concepts represented by nouns. The second component is a network of concepts represented by verbs and the third component includes descripitions of adverbs and adjectives. Along with these components, other linguistic information necessary for the text processing like a file of glosses is also included. Wordnet is an effort to combine traditional lexicographic information and high speed computing that rooted out of the university of Princeton under the supervision of George Miller in the early 90's. In fact, to date Wordnet is the most commonly used lexical repository for language processing tasks. The ontology underlying this semantic network is a graph where the nodes are concepts and the edges are relations. Some of the important relationships captured in the hierarchy include hyponymy, hypernymy, meronymy, holonymy and antonymy. An illustration of one of the top-most concepts with the synset **{entity, something}** from the noun taxonomy of the Wordnet semantic network is shown below :-

```
00001740 03 n 02 entity 0 something 0 014 ~ 00002086 n 0000
~ 00003095 n 0000 ~ 00003731 n 0000 ~ 00009457 n 0000 ~
03435902 n 0000 ~ 03495843 n 0000 ~ 03614902 n 0000 ~
06331805 n 0000 ~ 06683928 n 0000 ~ 06684175 n 0000 ~
06846327 n 0000 ~ 06847052 n 0000 ~ 06847350 n 0000 ~
06847525 n 0000 | anything having existence (living or
nonliving)
```

Every concept is indexed and identified by a unique sense number. Followed by the set of synonyms that represent the concept (lexical relationships of the concept), various semantic relations to other concepts are also represented. Every concept is further elaborated by a definition. Pointers to concepts that denote relationships to a concept in concern are identified by unique symbols. A list of such symbols used in the noun file include :-

! Antonym
@ Hypernym
~ Hyponym
#m Member meronym
#s Substance meronym
#p Part meronym
%m Member meronym
%s Substance holonym

Wordnet also provides index files which link every lexeme in the network to a list of all the senses that they can represent. The polysemy of each word is captured in these index files. To illustrate the span of this on-line lexical resource, a total of 94,474 lexemes are linked to 1,16,317 unique noun concepts, 10,319 lexemes are linked to 22,066 unique verb concepts, 20,170 lexemes are linked to 29,881 unique adjective concepts and 4,546 lexemes are linked to 5,677 unique adverb concepts.

Although Wordnet is the most commonly used on-line lexical resource for NLP tasks, it has many limitations. First and foremost, the structure is not built to facilitate even minimal forms of commonsense reasoning. Apart from inconsistent cycles that exist in the concept taxonomy, multiple inheritance is common between the concepts. Fig.14 illustrates multiple inheritance at the upper levels of the semantic network.

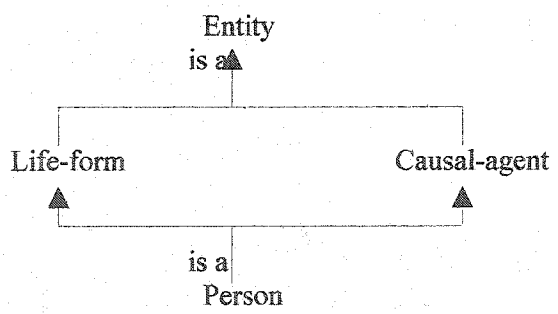


Fig 14: An instance of multiple inheritance in the Wordnet semantic network.

Yet another flaw with the conceptual structure of Wordnet is the lack of distinction between sub-types and roles. Fig.15 illustrates one such occurrence:-

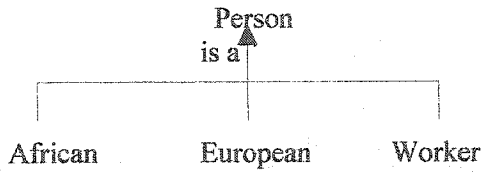


Fig 15: Confusion between roles and sub-types in the Wordnet semantic network.

African, European and Worker are roles that a person may play and should be treated in isolation from true hyponyms. The importance of treating roles differently from sub-types roots from different reasoning strategies that roles demand for. The study of roles is a broad domain in itself which we do not focus on in this survey.

4.3 The Mikrokosmos Ontology

The Mikrokosmos ontology (Mik, 1997), (Mahesh, 1995a) and (Mahesh, 1995b) is an integral part of a machine translation system currently under development at the computer research laboratory, New Mexico State University. It is a language neutral repository of concepts in the world to enable translations from one language to another. The ontology divides at the top level into object, event and property. Nodes occurring beneath these divisions in the hierarchy constitute the concepts in the ontology and are represented as frames consisting of slots with facets and fillers. Concepts have slots for a definition in a natural language, a time stamp, links to super/sub ordinate concepts and an arbitrary number of other properties. These slots have facets, each of which in turn has a filler. Facets capture such things as permissible semantic types or ranges for the slots denoted by (sem), the actual values if known denoted by (values) and the default values (default). An example illustration of a concept from the Mikrokosmos ontology is listed below :-

```
( MAKE-FRAME  ARTIFACT  (IS-A  (VALUE  (COMMON  INANIMATE ) ) )
  ( SUBCLASSES
     ( COMMON  ANIMAL-RELATED-ARTIFACT  AIRPORT-ARTIFACT
      ARTIFACT-PART  BUILDING-ARTIFACT  DECORATIVE-ARTIFACT
```

```
      DEVICE  EARTH-RESOURCE-ARTIFACT  ENGINEERED-ARTIFACT
      EVERYDAY-ARTIFACT  MEASURING-ARTIFACT  MEDIA-ARTIFACT
      MUSICAL-INSTRUMENT  PACKAGING-MATERIAL  PROTECTION-OBJECT
      PROTECTION-OBJECT  RESTAURANT-ARTIFACT  RESTRAINING-ARTIFACT
      SMOKING-DEVICE  VEHICLE ) ) )
  ( PART-OF  ( SEM  ( COMMON  ARTIFACT ) ) )
  ( DEFINITION
    ( VALUE
      ( COMMON  "physical objects intentionally made by humans" ) ) )
( AGE  ( SEM  (COMMON  ( >= 0 )  ( <> 0  20 ) ) ) )
( TIME-STAMP
  ( VALUE
    ( COMMON  "lcarlson at Monday, 6/4/90  12:49:46  pm"
      "updated by lori at 14:02:32 on 03/15/95"
      ---------- etc ---------- ) ) )
( COLOR
  ( SEM
    ( COMMON  RED  BLUE  YELLOW  ORANGE  PURPLE  GREEN  GRAY  TAN CYAN
MAGNETA ) ) )
( OWNED-BY  ( SEM  ( COMMON  HUMAN ) ) )
( MADE-OF  ( SEM  ( COMMON  MATERIAL ) ) )
( PRODUCT-TYPE-OF  ( SEM  ( COMMON  ORGANIZATION ) ) )
( PRODUCED-BY  ( SEM  ( COMMON  HUMAN ) ) )
( THEME-OF  ( SEM  ( COMMON  EVENT ) ) )
( MATERIAL-OF  ( SEM  ( COMMON  * NOTHING * ) ) ) )
```

The * NOTHING * is used to block irrelevant inheritance. The Mikrokosmos ontology has a span of roughly 4500 concepts with and an ongoing process to include more. A set of well defined rules and heuristics are adhered to while adding new concepts to the taxonomy. The primary motivation of Mikrokosmos being machine translation of natural languages, much damage can be done by multiple inheritance in the structure. This assumption is based on the notion that a certain degree of commonsense reasoning is necessary for machine translation. If so, multiple inheritance will disrupt any such inferences at the pragmatic level. On an average, every concept in this taxonomy has up to 4 parents other than itself. It is hard to imagine how one could build up to 3 to 4 levels of inferences with a structure that has so many contradictions by default. Fig.16 illustrates two occurrences of mulitple inheritance at the upper levels of the structure. It is important to note that these instances occur at the upper levels of the ontology. We emphasize this location primarily because the higher up in the hierarchy, the more severe are the consequences of multiple inheritance.

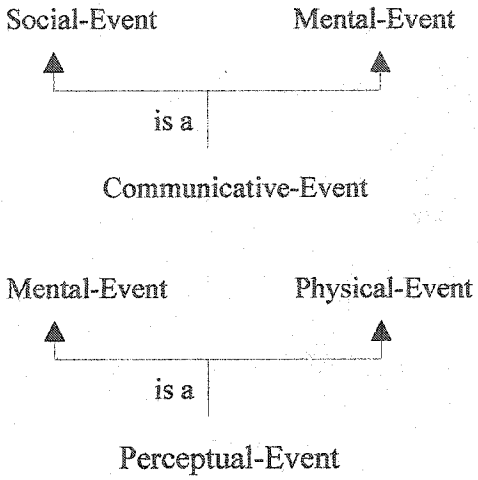


Fig 16: Occurences of multiple-inhritance in the Mikrokosmos ontology.

4.4 Other Ontologies

Two other ontologies one is bound to encounter when working in NLU/NLP are the Generalized Pennman Upper model and the EDR. The Pennman Upper Model is a consequence of work done in language generation at ISI in the 1980's (Bateman et al., 1994). It emerged as a general and re-useable resource, task (domain) independent semantic classification. The analysis of semantic distinctions in this repository was achieved through the alterations of grammatical variations in and across languages. LOOM, a knowledge representation language developed at ISI was used to represent the ontology underlying this model. The original Pennman Upper Model merged with the KOMET German Upper Model to then become the Generalized Pennman Upper Model which is how it is currently referred to.

On the other hand, the EDR is yet another machine translation effort that aims at a complete language neutral conceptual classification. The aim behind the development of EDR was translation of texts from English to Japanese, and vice versa. The EDR approach to classification is different from other known approaches primarily because the semantic network is a mediation between grammatic rules and templates used in, and across English and Japanese. Although similar to Wordnet in the commitments of the

ontology, fewer relations are examined. The relations accounted for by the EDR are restricted to synonymy, hyponymy and entailment. As a consequence, the English lexicon and the Japanese Lexicon are being developed in parallel.

Although all ontologies have their own differences in the end goals and approaches they follow, the common understanding that language is the key to organizing knowledge is shared by all of them, lesser with Cyc and more with the others. Amongst all the existent knowledge structures, the CYC knowledge base is clearly a step ahead in this race towards commonsense reasoning. It is the closest we are to making computers perform some of the basic cognitive activities that even 10 year olds perform sub-consiously. CYC has gone that extra step in moving up to the knowledge level where current efforts are focused on smart algorithms to infer from the world knowledge available in their repository.

Chapter 5

Digital Agora: An Intelligent Text Processing System

The text available to us on the electronic highway seems to be increasing at an exponential rate. With more and more information pouring in day by day, there seems to be too much information around us, and not sufficient time to sort through it. Moreover the state-of-the-art search technology does not seem to help much either, primarily due to its incompetence in exploiting this rather rich pool of information. Unlike in the early day where most information was stored in databases, and structured queries based on well founded algebra's were used to retrieve this data, what we see around us today is unstructured text in the form of journals, email messages, news briefs, and so on. The lack of structure in these documents poses a hurdle to effectively searching, sorting, classifying, indexing, retrieving and filtering them. There seems to be no other way to process such forms of unstructured data without having to either interpret or understand it. This need has shifted the focus of information retrieval (IR) from that of traditional database access to that of NLP based text interpretation.

5.1 NLP-based Information Retrieval

Information retrieval primarily deals with the task of retrieving documents relevant to a given query. The goal of an information retrieval system is to locate relevant documents in response to a user query (Krovetz et al., 1992). Most IR systems usually rank retrieved documents in accordance to their levels of relevance to the user query. This relevance is a similarity measure of the document to the query. The performance of an IR system is measured by the precision and recall of its retrieval. Precision is the number of relevant documents retrieved over the total number of retrieved documents, whereas recall is the number of relevant retrieved documents over the total number of relevant documents in the system. These measures are computed as a consequence of the performance of the system when applied to standard test set collections that provide a set of queries, a set of documents, and a set of relevance judgments to indicate the relevancy of each document to each query.

Traditional IR systems represent documents and queries by the words they contain (bags of words), and the similarity measure is usually some function of the common words appearing in the query and the document. The more words the query and the document have in common, the higher is the relevance. Such key-word systems have rather low precision and recall, primarily due to the following problems: (1) ambiguity occurring in words can cause irrelevant documents to be retrieved, (2) relevant documents that contain synonyms and related words are ignored, and (3) change in subject caused due the combination of lexical nominals. Other variations of IR approaches include sense-based retrieval, and subject-based retrieval. In sense-based retrieval, IR systems represent, index and retrieve documents based on the disambiguated senses of the key-words occurring in the document (bags of senses), whereas subject-based retrieval approaches involve representing, indexing and retrieving documents based on topic summaries extracted from the various documents. Although sense-based retrieval acts as a solution to limitations (1) and (2), when compared to the traditional bags of words in improving the precision and recall of searches, to address limitation (3), we require more powerful

subject-based retrieval. Key-word based retrieval is purely syntactic in nature, whereas both bags of senses, and subject-based retrieval require various levels of semantic and pragmatic computation. The levels of semantic and pragmatic computation needed for each of these approaches remains the same up to a point where words have to be disambiguated under a given context, and differ in that subject-based retrieval demands for a powerful algebra that allows us to compose, decompose and make sense of semantic entities. The distinction between these various approaches to IR is best illustrated in fig.17.

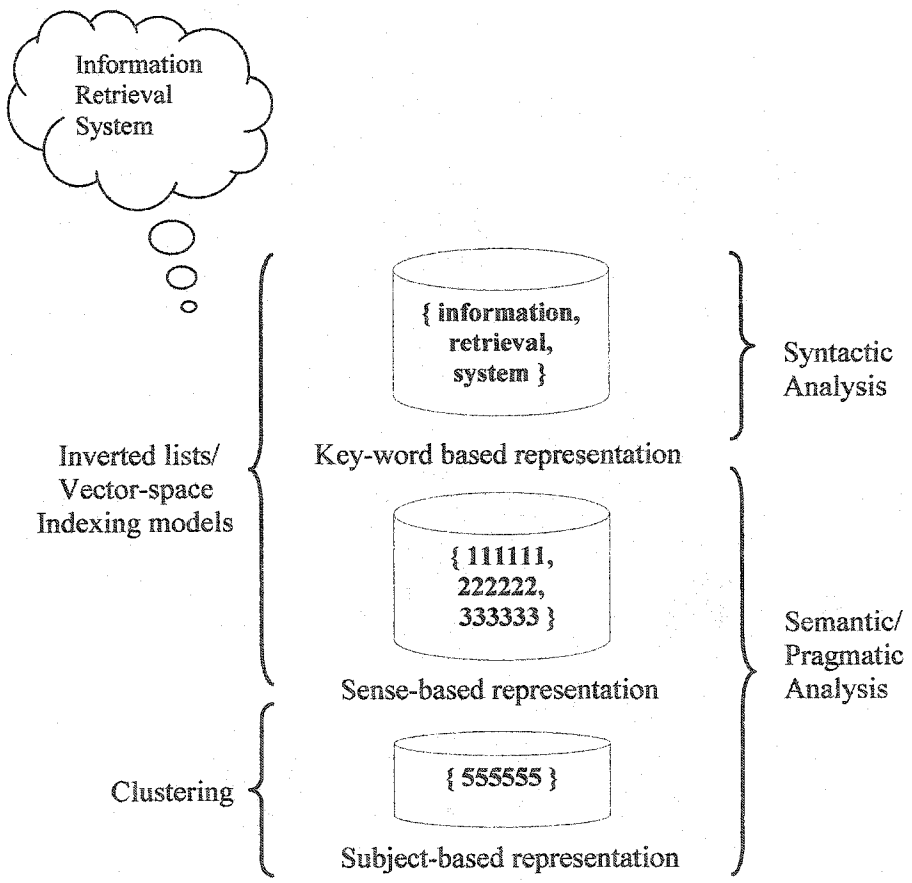


Fig.17: Representations for a given document and the required computation models.

As shown in fig.17, a document can either be represented as bags of words, bags of senses or by a concept that denotes the subject summary of the document. The phrase

information retrieval system is represented by the vector **{information, retrieval, system}** in a key-word based system, and as the vector **{1111111, 2222222, 333333}** in a sense-based approach. Here the elements of the vector represent the sense number for the disambiguated meaning of each of the respective tokens. In a subject-based approach however, the document is represented by **{5555555}**, where the elements of the vector represent compound concepts that comprise the content summary of the document. These compound concepts are a consequence of the algebra being applied to the disambiguated senses of **information**, **retrieval** and **system** respectively. As mentioned earlier bags of words require only syntactic analysis, whereas bags of senses, and subject-based representations require additional semantic and pragmatic reasoning. Needless to say, subject-based document representation demands the most complex computation when compared to the other two approaches, as it involves interpretation of the document in building the summary. Moreover, even the indexing model that underlies a subject-based retrieval model is more expensive when compared to the traditional inverted index, or the vector space model, commonly used with bags of words/bags of senses. Primarily so, because words and the various senses that these words denote (primitive concepts) are finite, whereas the set of derivable compound phrases that constitute the summary are infinite. According to Wordnet, there are 94,474 unique nouns in English, and 1,16,317 unique senses that these words possibly represent. Therefore, a document can be indexed by the words/senses that represent the document.

The lack of such a finite set of keys when indexing documents based on subject summaries makes it necessary to match a given query with all the document summaries in the repository, thereby making both the inverted-index, and the vector-space models inappropriate for the purpose. Therefore, the more feasible approach to indexing/retrieving documents in such a model is to narrow the search space for given queries by pre-clustering the documents in the system. The goal of document clustering is to create/discover a reasonable set of clusters for a given set of documents. A reasonable cluster is one that maximizes with-in cluster document similarity, and minimizes between-cluster similarity. Therefore, if a collection of documents are categorized into **n** conceptually different clusters, then queries could first be compared against

representations of each of these clusters. Once a smaller subset of these clusters are selected as possible candidates, every document in these clusters can be then matched with the query to determine relevant documents. Apart from the improved efficiency of cluster-based retrieval with ad-hoc document collections (such as topic-based retrieval systems), the cluster hypothesis further adds that retrieval from a cluster-based collection improves recall and precision (Jurafsky et al., 2000). Primarily so, because if documents are pre-categorized based on topic similarity, then it is the assumption that irrelevant documents will be avoided completely in the retrieval process. Therefore the likelihood of retrieving irrelevant documents is low.

As solutions that improve search precision and recall, a substantial portion of the IR community is moving towards more sophisticated techniques that use subject-based retrieval. In such approaches, not only is it necessary to formulate suitable semantic representations for documents, but it is also necessary to describe an algebra that can be applied to the semantic entities (both primitives and composites). In such computational models, documents and queries are identified by semantic objects that represent content, rather than key-words or mere senses. Also commonly known as topic-based information retrieval, such approaches demand the use of large linguistic semantic networks that enable such forms of semantic/pragmatic computation. Although continual efforts have been in progress to build such systems, nobody has been successful as of yet. The problem remains in that such approaches prove computationally expensive, due to the lack of formal solutions to the various ambiguities in language.

The experiment we describe here is one such effort to intelligent text processing, where we modeled and implemented a NLP based text-retrieval system (Digital Agora) that indexes, searches and retrieves documents based on subject. The focus of this chapter is to describe the experiment (model and process that underlies Digital Agora), and identify the various pitfalls that hampered its completion into becoming an efficient/effective solution to content based text retrieval. Some of the immediate problems that we shed light upon include (1) a computationally expensive retrieval model that Digital Agora demands for, (2) the computational ineffectiveness and inefficiency of the involved

lexical disambiguation process, (3) the lack of a formal algebra to extract the meanings of compound-nominals, and (4) time vs quality tradeoffs when parsing. Apart form discussing some of the important heuristics used to find moderate practical solutions in overcoming these problems, we conclude by identifying the need for better semantic networks (when compared to existing ones), that provide richer and well-structured semantic information.

5.2 The Goal of Digital Agora

The goal of a topic-based IR system is to index, classify and retrieve documents based on subject content. Unlike traditional search technologies that use keywords as document representations, topic-based IR systems need to interpret, understand and summarize a document. This summary, a good representation of the document content, is then used in indexing and retrieval. In subject-based searches, there is very little processing to be carried out at the syntactic level. Document-query relevance is computed at the semantic level, where the words themselves act as mere symbolic representations of meaning objects, and it is these semantic counterparts that are used in the computational process. The computational process underlying Digital Agora is one such form of partial semantic language analysis that aims to extract key topics from given documents to form respective representative summaries. Unlike traditional search technologies (bags of words), this system aims to conceptually match topics like **document management system** and **information retrieval engine**. Although the key words in each of these topics are entirely different, these topics are really about the same thing. When we are searching for **document management systems**, for example, a document that is about **information retrieval system** should be retrieved, and vice versa. On the other hand, it is also important to differentiate between topics like **insurance company** and **company insurance**, because although the exact same key words appear in both phrases, they are actually talking about two entirely different topics. When the search topic is **insurance company**, it is most likely that we are interested in the company that provides insurance rather than a specific policy. On the other hand, when we talk about **company insurance**,

we are most probably interested in the insurance policy for a company rather than the insurance company itself. Although the difference here may seem rather subtle, the sentences listed below illustrate this better:

1) eat what you see ≠ see what you eat.
2) live to eat ≠ eat to live.

The lack of ability in traditional key-word based search technologies to account for such similarities/dissimilarities hampers the effectiveness (i.e. precision) of their retrieval. Traditional search technology offers at most a 40% precision and recall when retrieving documents from open text. This is so because such deficiencies add up to an exponential deviation in the end result. The goal of Digital Agora is to compute document relevance at this semantic level. Needless to say at this level of computation, both the precision and the recall of searches show significant improvement. But sadly enough, this added benefit has its own costs, computational complexity, an issue addressed as a continuous theme throughout this chapter.

5.3 The Functionality of Digital Agora

Like any other topic-based text-retrieval application, the basic functionality of Digital Agora can be broadly classified into two parts: (1) index documents to be stored in the repository, and (2) retrieve and rank documents relevant to a given query. The difference however lies in how we index and retrieve these documents. Unlike traditional IR systems, in Digital Agora, a document is represented by a set of key topics that represent the content summary of the document. The indexing phase involves interpreting, summarizing and storing incoming documents into a clustered repository, whereas the retrieval phase involves interpreting a query, and matching the query with documents from selected clusters to retrieve and rank relevant documents. We use the seed based clustering approach to divide our repository into sub-categories. We chose a pre-defined set of 100 broad subject categories, the kind of which one finds in the taxonomies of web search sites. After the key subject(s) of a document have been identified, documents are

placed in the right cluster. These categories were selected from the yellow pages of Yahoo, and included topics such as **computer-science**, **automobiles**, **politics**, **business**, **sports**, and so on. Although other approaches such as the "scatter-gather" approach to document clustering may improve quality of the clusters, we decided to settle with a pre-defined list of categories because: (1) other clustering algorithms have a rather high complexity – $O(n^2)$, and (2) with all other clustering algorithms, there is a need to re-cluster the entire repository whenever new documents are added (because these clusters are relative to the entities in the collection). Moreover, as the goal of the experiment was more focused towards document representation, and summary-extraction, seed-based clustering was sufficient in providing us with the required framework. However, without the loss of generality, given a realistic situation where the search engine actually sits on an alpha-deck server, it is practical to substitute the seed-based clustering algorithm with any other form of clustering, whereby the clustering process would become a low-priority back-end process that occupies the processor at low traffic times when the server is not handling any search requests. The functionality of Digital Agora is best illustrated in fig.18:

As illustrated in fig.18, steps I-1 to I-4, illustrate the indexing process, and steps R-1 through R-6 illustrate the retrieval process. As mentioned earlier, every one of the clusters in the repository is denoted by a pre-defined topic. Therefore, once an incoming document is summarized, the summary of the document is then matched to every one of the 100 pre-defined cluster topics. The cluster denoted by the topic with the strongest match to the document summary forms the category that will house the document. Similarly, the semantic representation of a given query is matched to the topics of the clusters. The cluster represented by the topic with the maximum match to the query is the pool from which all document summaries will be matched against the query to determine relevant documents. In the next few sub-sections we describe the parse, the semantic-representation phase, the object-model used to represent documents and the topic-extraction process. Along the way we also address the various limitations encountered, and discuss various heuristics and trade-offs implemented to overcome some of these limitations.

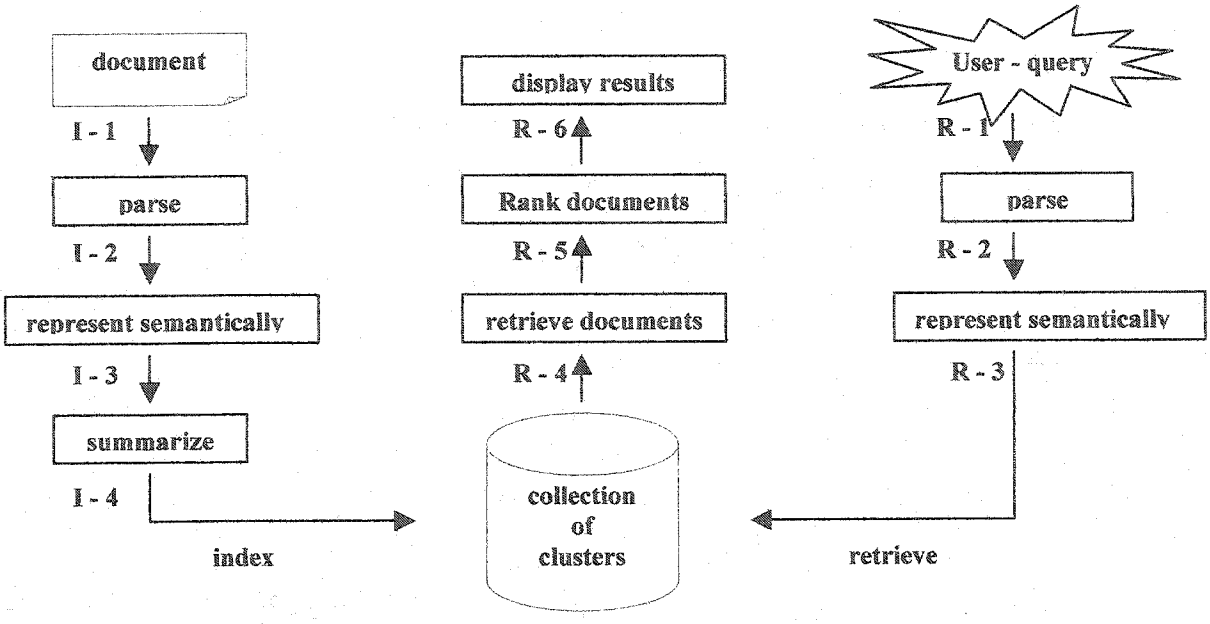


Fig.18: Functionality of Digital Agora.

5.4 Syntactic Analysis

5.4.1 Parsing and Generating Syntactic Objects

Parsing is the process of mapping a text to syntax (or derivation) trees, i.e. the task of recognizing a phrase and assigning a syntactic structure to it (Jurafsky et al., 2000). Most language parsers are based on variations of Context-Free Grammars (CFG) that determine a set of production rules to either break down a given sentence into its constituents, or build up the composite sentence from the set of constituents. Commonly known as Top-down parsing and Bottom-up-parsing, these parse techniques are commonplace in NLP/NLU systems. In Digital Agora, we have implemented a simple top-down recursive descent parser. The goal being that of extracting key topics, we were not interested in complex syntactic analysis, but rather in nominal processing, which we called TPhrase (term phrases). For example, we were simply interested in phrases such as **information retrieval systems, the application of machine learning in text retrieval,** etc. Note that phrases can include prepositional phrases, although their semantics were

not fully developed (more on this below.) The CFG implemented by the parser in Digital Agora is shown below:

```
// Non Terminal Symbols
TPhrase  ::= Tp  Preposition  TPhrase
           |  Tp

Tp  ::= Adjs  CNouns
      |  Cnouns
      |  Pnouns

Adjs  ::= Adj  Adjs
        |  Adj

CNouns  ::= CNoun  CNouns
          |  CNoun

PNouns  ::= PNouns  PNoun
          |  PNoun


// Terminal Symbols
Adj  ::=  artificial | solid | angry | etc.        // adjectives looked-up in Wordnet
CNoun  ::=  computer | screen | dog | etc.        // nouns looked-up in Wordnet
PNoun  ::=  socrates | usa | paris etc.           // names of peoples, places, etc.[1]
```

According to the CFG listed above, a TPhrase can be either a Tp, like **computer science building**, or Tp Preposition Tp like **computer science building in the University of Windsor**. Similarly, a Tp can either be a list of zero or more adjectives followed by a common-noun like **boy**, **tall boy**, **tall handsome boy**, and so on, or a Tp can be a list of proper-nouns like **Jack**, **Jack Henderson Smith**, and so on. As the goal of the system was to identify the key-phrases that represent the subject content of the document, a shallow parser built around this CFG was sufficient. Under the assumption that phrases occurring in the document are sufficient to denote the subject content of the document, the CFG ignores verbs/adverbs to facilitate efficient shallow parsing. Moreover, as there is no connection specified in Wordnet between nouns and verbs/adverbs that aids in the

[1] We actually extended Wordnet, which did include some names, with specific product, company, media names, etc.

semantic computation, ignoring verbs/adverbs increased the efficiency of the parse without reducing the effectiveness of the semantic processing that follows.

As the implementation of Digital Agora was based on the Object-Oriented approach, at the syntactic level, every object was implemented as a subtype of a LingObj. The parser was built as an interaction (message passing) between these various linguistic objects. Every non-terminal occurring in the CFG was implemented as a sub-type of this LingObj. The relation between these various linguistic (syntax level) objects implemented in Digital Agora are best illustrated in the class diagram listed below:

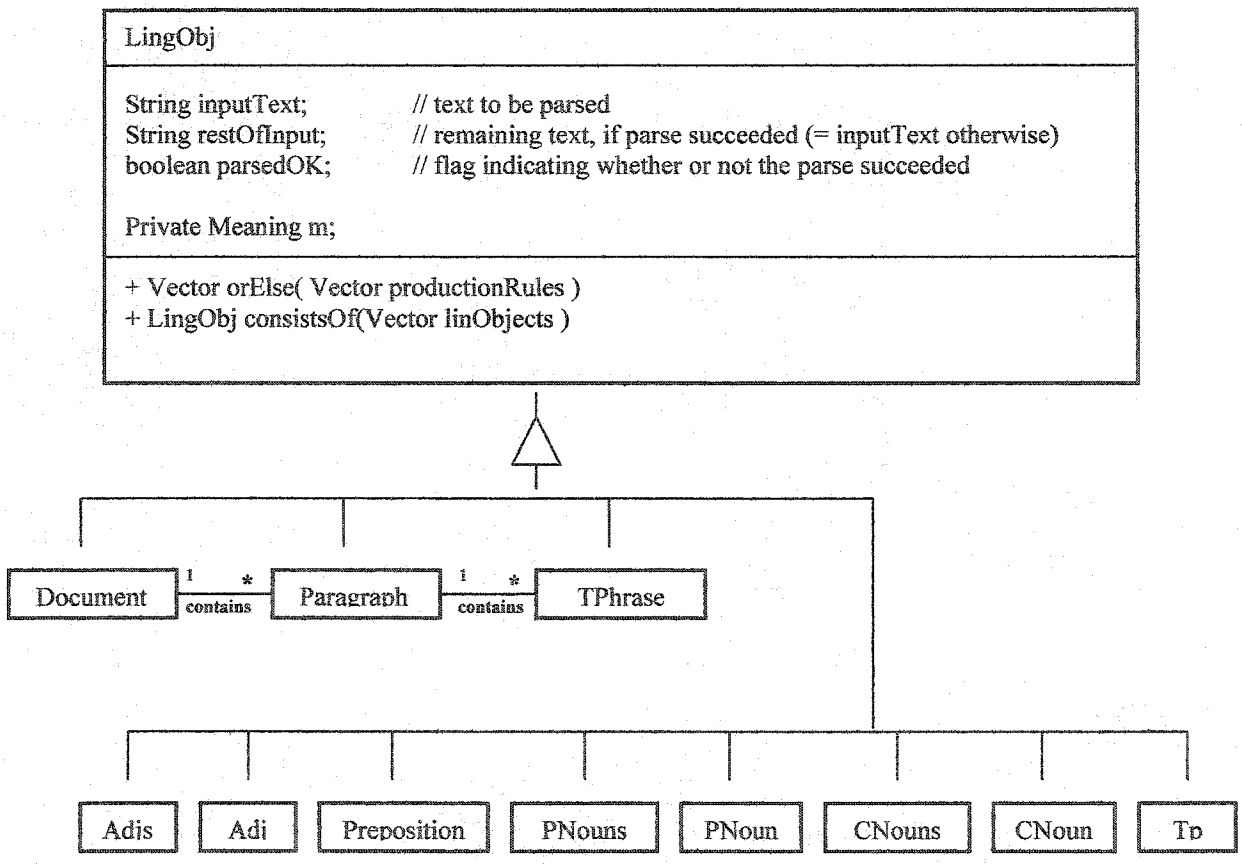


Fig.19: Object diagram of the syntactic objects.

The largest linguistic object (which is a composite of all the other linguistic objects) is actually the document itself. As illustrated in the class diagram above, a document may

contain one or more paragraphs, and a paragraph may contain one or more TPhrases, where a TPhrase represents a possible multitude of combinations of nouns, adjectives, prepositions and separators respectively (as defined by the production rules of the CFG). The constructor of the document takes as textual input the entire file. The file is then split into several paragraphs (using paragraph tags, and , which was added in a pre-process). The paragraph, on the other hand, contains a set of TPhrases that have been parsed properly. Much rather than treating a document as a collection of phrases, we place these intermediate paragraph objects because the topic extraction algorithm, as discussed in the later sections, assumes such a breakdown to measure the strength of topics across the paragraphs (i.e a topic that occurs in more than one paragraph is given more weight than a topic that occurs only in one paragraph). Moreover, as illustrated in Fig.19, every LingObj contains its semantic counterpart (the Meaning object). These meaning objects represent the value of the respective syntactic object. The instantiation of these semantic objects takes place once the parsed terminals/non-terminals are combined to form other non-terminals (more about this when we discuss semantic anlaysis in the section 5.5).

The top-level linguistic object (LingObj) contains all the parsing functionality, most importantly are the two combinators "consistsOf" and "orElse" which correspond to the BNF notations "::=" and "|". Since our implementation was done in OO language, we followed a "parse by the object construction" strategy. Basically the approach is as follows:

- For every terminal and non-terminal symbol a class is defined as a subtype of LingObj.
- The constructor of the class is given the text to be parsed (this is the inputText).
- The parse( ) method is then called.
- If the parse succeeded, the remaining text (if any) is placed in restOfInput.
- The flag parsedOK is set accordingly.

To understand this process better, let us analyse the simple parser/class of a CNoun Object (for common nouns).

```
1)  class CNoun extends LingObj
2)  {
3)          CNounConcept        m;
4)          String              word = new String("");
5)
6)          CNoun() {}
7)          CNoun(Vector words) { text=words; }
8)          CNoun(Vector words, Vector context)
9)          {
10)                 inputText = words;
11)                 word = inputText.elementAt(0);                  // pick-up the first token
12)                 if ( isNoun( word ) )
13)                 {
14)                         restOfInput  = words.removeElementAt(0);    // throw first word
15)                         parsedOK = true;
16)                         m  =  new CnounConcept(word, context);      // populating semantic object
17)                 }
18)                 else
19)                 {
20)                         restOfInput  = words;                       // no text was processed
21)                         parsedOK = false;
22)                 }
23)         }
24) };
```

Fig.20: The Constructor (implementing the parser) for a CNoun object.

As shown in Fig.20, the parse for a CNoun simply involves checking to see if or not the given token (word) under the given context is a common noun. The function **isNoun(word)** in line (12) does just that. If the given word is parsed as a common noun, the rest remains to remove this token form the initial set of tokens to form a new **restOfInput** that is yet to be parsed. Other terminal symbols (e.g., adjectives and proper nouns) have a similar structure. The parser\class for non-terminal symbols have a slightly more complex functionality. To illustrate one such scenario, let us take a look at the parser for the Tp (term phrase) class (fig.21).

As with the case of other classes that represent non-terminal symbols, the parser of the Tp object (fig.21), calls the **xOrElse( )** function, passing as arguments, itself, the text to be parsed, and the set of production rules permissible to form a Tp object (line 16). The **xOrElse( )** ("|") function, in turn, calls the **consistsOf( )** function ("::=") iteratively to determine a parse for the given Tp. The result of the **xOrElse( )** function (line 16) is a vector with two elements. The first element is an integer that represents the rule that was applied, and the second element is the parsed LingObj itself. This information (the rule) is necessary when generating the necessary semantic objects. Lines(22) to (25) deals with

creating these semantic counter parts, and are discussed in more detail in section 5.5 on semantic analysis, where we analyze the TpApply( ) function.

```
1)  class TP extends LingObj
2)  {
3)          TPConcept           tpMeaning;
4)
5)          TP() {}
6)          TP(Vector words) {super(words);}
7)          TP(Vector words, Vector context) {super(words, context);}
8)
9)          public void parse()
10)         {
11)                 CNouns cns = new CNouns(text,context);
12)                 PNouns pns = new PNouns(text,context);
13)                 Adjs adjs = new Adjs(text,context);
14)
15)                 // this method returns once a rule succeeds (xOrElse as opposed to OrElse)
16)                 Vector results = xOrElse(this, text                    // input text
17)                                         , rule(adjs, cns)              // first rule to try
18)                                         , rule(cns)                    // second rule to try
19)                                         , rule(pns)                    // third rule
20)                                         );
21)
22)                 // steps involved in populating corresponding semantic objects
23)                  int ruleNo  =  ((Integer) results.elementAt(0)).intValue();
24)                  Vector rule  =  (Vector) results.elementAt(1);
25)                  tpMeaning  =  TpApply(ruleNo, rule);
26)         }
27)  };
```

Fig.21: The Constructor (implementing the parser) for a Tp object.

Note that since we have not included the semantic processing methods, the context parameter that is passed to the constructor is currently ignored (this will be used when we show how the semantics are performed in tandem with the parsing). We were able to completely ignore the context during the parse by pre-tagging the document with POS tags as illustrated in the following section.

5.4.2 Part of Speech Tagging (POS)

Initially, we had a somewhat complete parser in that we tried to find out the part-of-speech tag for words by trying different parses. For example, for our parser to figure out that **break** in **john had a long break** was a noun, as opposed to a verb, our parser had to try a number of rules. If the text was tagged however, the parser simply assembles components without trying these various rules.

Therefore, we attached part-of-speech (POS) tags to tokens in the document prior to parsing. We used a readily available off-the-shelf tagger (Brille), to pre-tag the document with the POS tags. Brille is an application that attaches POS tags to the tokens in a document based on fairly standardized heuristics formulated from statistical corpus analysis. We chose to separate this functionality away from the parser, as Brille provides for highly effective/efficient POS tagging. Its effectiveness is in that the end product is close-to-perfect, and its efficiency is in that, apart from applying a lesser number of well-formulated rules than our parser initially did, it also minimized dictionary look-ups (look-ups to data-structures that hold the dictionary), when determining word categories, by using heuristics based on statistical evidence. Moreover, it relieved us the responsibility of building into the parser intricate details of POS disambiguation.

5.4.3 Pre-Parse Lexical Analysis

The parsers implemented in the various Lingobj's assume that there are functions (like the **isNoun(car)**), that perform lexical look-up from our dictionary (which was that of Wordnet). This process involved two challenges: (i) there was the issue of morphology (that is, how **students** will be looked-up, where the dictionary only had the root word **student**), and (ii) combining several tokens that existed in the dictionary as one token (e.g., **artificial** and **intelligence** had to be combined to look-up **artificial_intelligence**, a single token in Wordnet).

Commonly known as morphological stemming, the process of deriving the stem from inflected/derived words, applies to many affixes other than plurals, like **type** from **typing**, **dinner** from **pre-dinner**, and so on. Even with plurals alone, classes of words follow different rules, like for example, the plural of **fish** is **fish**, the plural of **car** is **cars**, whereas the plural of **church** is **churches**. In each of the listed examples, the suffix we add to get the plural is different. However, it is not always the case that the stem is unaltered in the inflicted form, like with the case of **factories**, where the stem is actually **factory**. Such forms are treated as exceptions, and most electronic dictionaries usually supply a list of words that have exceptional morphological rules. Wordnet comes

equipped with such a morphological exception file for each of the POS categories. Therefore, upon following a set of general rules and using these exception files, the process of stemming became a rather trivial implementation.

In Digital Agora, we implemented a morphological parser to retrieve root words for nouns and adjectives. As an efficiency/quality trade-off, we restricted the stemming only to that of removing suffixes occurring with nouns/adjectives. Primarily, because suffixes are used rather rampantly in language, and cannot be ignored, whereas prefixes, on the other hand, are seldom occurrences, and can therefore be ignored without the loss of much content. The **getRoot( )** method implemented by Digital Agora is listed below:

```
1)  public String getRoot(String tok)
2)  {
3)          String root = (String) nounMorphs.get(tok);
4)
5)          if( root == null ) root = (String)adjMorphs.get(tok);
6)          if( root == null ) root = applyNounSufixes( tok );
7)          if( root.equals( tok ) ) root = applyAdjSufixes( tok );
8)
9)          return root;
10) }
```

Fig.22: The getRoot( ) function for morphological analysis.

As used in fig.22 (lines (3) and (5)), **nounMorphs** and **adjMorphs** are Hashtables that represent morphological exceptions occurring with nouns/adjectives respectively. The keys of the Hashtables are the inflicted forms, and the elements are the root-words. Therefore, given an input word, **getRoot( )** simply checks to see if these inflicted forms are found in the lists of exceptions. If they are found, the root word is retrieved from the list and returned, and if not then either of **applyNounSufix( )** or **applyAdjSufix( )** is called (lines (6) and (7)). As mentioned earlier, the two general classes that need to be taken care of by the **applyNounSufix( )** (plurals for nouns) are those inflicted words that end with suffixes "es", and "s", and similarly those taken care of by **applyAdjSufix( )** (derivations of adjectives) include those inflicted forms ending with "er" and "est". Listed below are the functions **applyNounSufixes( )** and **applyAdjSufixes( )** that implement this functionality.

```
1)  public String applyNounSufixes( String token )
2)  {
3)          int tokenLength = tokenLength = token.length();
4)
5)          if( token.endsWith("es") )
6)          {
7)                  String root = token.substring(0, tokenLength - 2);
8)                  if( wn.index.get( root ) != null )
9)                          return root;
10)         }
11)
12)         if( token.endsWith("s") )
13)         {
14)                 String root = token.substring(0, tokenLength - 1);
15)                 if( wn.index.get( root ) != null )
16)                         return root;
17)         }
18)
19)         return token;
20)  }


21)  public String applyAdjSufixes( String token )
22)  {
23)         int tokenLength = tokenLength = token.length();
24)
25)         if( token.endsWith("er") )
26)         {
27)                 String root = token.substring(0, tokenLength - 1);
28)                 if( wn.adjex.get( root ) != null )
29)                         return root;
30)
31)                 root = token.substring(0, tokenLength - 2);
32)                 if( wn.adjex.get( root ) != null )
33)                         return root;
34)         }
35)
36)         if( token.endsWith("est") )
37)         {
38)                 String root = token.substring(0, tokenLength - 2);
39)                 if( wn.adjex.get( root ) != null )
40)                         return root;
41)
42)                 root = token.substring(0, tokenLength - 3);
43)                 if( wn.adjex.get( root ) != null )
44)                         return root;
45)         }
46)
47)         return token;
48)  }
```

Fig.23: The applyNounSufixes( ) and the applyAdjsufixes( ).

As shown in fig.23, between lines (5) to (10), the suffix is eliminated for nouns ending with "es", to retrieve the root, and similarly between lines (12) to (17), the root word is derived by removing the suffix for those inflicted forms that end with "s". It is important to note that we first make sure it as an inflicted form before chopping of the suffix by checking to see if the word is present in the dictionary (lines (8) and (15)). Therefore, in

brief, the implemented rule is as follows: if the noun ends with either an "es" or an "s", and is not found in the dictionary, then we assume it is a plural, and eliminate the suffix to derive the stem. In a similar fashion, we remove suffixes for adjectives that end with "er" (lines (25) to (34)), and those that end with "est" (lines (36) to (45)).

Unlike the simple implementation of the morphological analyzer, we had come up with a non-trivial implementation to overcome limitation (ii), i.e., to facilitate the retrieval of tokens represented by compound-words like **computer_science**, **artificial_intelligence,** and so on, from the dictionary. The primary problem here was that given a sequence of **n** words, we had to do $O(n^2)$ dictionary look-ups to pick the largest compound nominal occurring in the dictionary. **Million_floating_point_operations_per_second** is one such compound multi-token entry in Wordnet. To make sure that we pick-up such a compound token, we have no choice but to check if this compound token occurs in the dictionary. In this case we were fortunate that it did, and therefore picked it up and terminated the process. But on the other hand, given the phrase, **artificial intelligence systems at University of Windsor**, to pick-up **artificial_intelligence** as one token, we have no option but to apply brute force checking for all combinations of the phrase, since we know that 6-word tokens like **million_floating_point_operations_per_second**, exist in the dictionary. The various look-ups necessary to identify **artificial_intelligence** as one token from the given phrase include:

1) artificial_intelligence_systems_at_University_of_Windsor
2) artificial_intelligence_systems_at_University_of
3) artificial_intelligence_systems_at_University
4) artificial_intelligence_systems_at
5) artificial_intelligence_systems
6) artificial_intelligence_systems
7) artificial_intelligence

Therefore, in the worst case, given a phrase with **n** tokens, we require **(n * (n+1))/2** dictionary look-ups to ensure that all maximum compound tokens occurring in the phrase have been captured. Primarily, because in the worst case, if we failed to find any compound tokens that start with **artificial**, the same process has to be repeated to check

for compound tokens that start with **intelligence**, and so on. For example, let us assume that **artificial_intelligence** did not occur as a compound token, but on the other hand, **intelligence_systems** exists in the lexicon as one token. In such a scenario, to identify **intelligence_systems** as a single entry, the same process has to repeated to find maximum compound tokens that start with **intelligence**. Therefore, with a complexity of $O(n^2)$, this process proved rather inefficient.

In an effort to avoid this added complexity, we conducted tests to see if ignoring such compound tokens would affect the quality of the resulting summaries. To our surprise, we found significant differences in the quality (effectiveness) of the results (at the semantic level). In brief, what we initially decided to do was retrieve only single token entries from the dictionary and then apply the algebra to form compound concepts (basically ignore multi-token dictionary entries). But to our misfortune, we realized that if such a multi-token entry is represented in the dictionary, it is much richer and more precise than the ones formed as a consequence of the algebra. Moreover, such compound concepts deemed very important phrases in the final summary, and thereby had the power to bias the entire summary. We explain this bias in greater detail when we discuss the topic-extraction algorithm in the later sections. Moreover, as shocking as it may seem, there are 47,173 such tokens in Wordnet. Roughly accounting for half the dictionary, this is a significantly large number to ignore. The largest such multi-token entry in Wordnet is **Economic_commission_for_asia_and_the_far_east**. Therefore, in order to achieve the desired precision accuracy of the summaries, we had no choice but to capture at-least most of these multi-token compounds.

In an effort to improve the efficiency of the process, we analyzed the following: (1) the multi-token entries in Wordnet, and (2) the occurrences of multi-tokens in a random set of 100 documents (from the corpus of the Ottawa Citizen). In checking the entries of Wordnet, we found that a large chunk of the multi-token entries are either entries with two tokens, or entries with three tokens. In fact we observed that two/three token entries accounted for roughly 90% of all the multi-token entries in Wordnet. The split-up of multi-token entries in Wordnet is as follows:

- Entries with two tokens: 42223
- Entries with three tokens : 4251
- Entries with four tokens : 555
- Entries with five tokens : 108
- Entries with six tokens : 27
- Entries with seven tokens : 5
- Entries with eight tokens : 4

Moreover, we also studied several news briefs from the Ottawa Citizen to find that those entries in Wordnet that have more than three tokens are very rare occurrences in free text. Therefore, as a trade-off on quality for speed, we decided to ignore multi-token entries that have more than 3 tokens. To our advantage, we had to sacrifice negligible precision for a rather considerable decrease in the required processing time, from that of **(n*(n+1))/2** to **3n** in the worst case, i.e, decrease in algorithm complexity form $\mathbf{O(n^2)}$ to **O(n)**. Other heuristics used to further reduce the required processing included the following:

1. Ignore tokens that begin with prepositions such as "in", "out", "near" "from", and so on.
2. Ignore tokens that begin with stop-words such as "when", "where", "how", and so on.
3. Ignore tokens that begin with adverbs.
4. Ignore tokens that begin with verbs.

These heuristics are appropriate, because compound-tokens in Wordnet begin either with nouns or adjectives. For example in the compound token **electrical_engineering**, the first token **electrical** is an adjective, and so on. The function **formulateConcepts( )** that implements this piece of logic is illustrated in fig.24:

As illustrated in fig.24, the **formulateConcepts( )** function takes as its argument a vector **qryVec** (line (1)), that contains all the Words in the document followed by their POS tags. As a result of POS tagging, in **qryVec**, the consequent element to a document word, is a string that represents its respective POS tag. For example, if the first element of **qryVec** is **computer** (the first word in the document), then the second element of this vector must be its tag (**nn**, a tag that represents a noun), and so on.

```
1)  private Vector formulateConcepts( Vector qryVec )
2)  {
3)
4)          Vector result = new Vector();
5)
6)          for( int i=0; i<qryVec.size(); i+=2)
7)          {
8)
9)                  String tok1 = (String) qryVec.elementAt(idx);
10)                 String tok2 = (String) qryVec.elementAt(idx + 2);
11)                 String tok3 = (String) qryVec.elementAt(idx + 4);
12)
13)                 String tag1 = (String) qryVec.elementAt(idx+1);
14)
15)                 If( adjCodes.get(tag1) != null  || nounCodes.get(tag1) != null )
16)                 {
17)
18)                         String threeWord = tok1 + "_" + tok2 + "_" + tok3;
19)                         String twoWord = tok1 + "_" + tok2;
20)
21)                         if( wn.index.get( threeWord ) != null )
22)                         {
23)                                 result.addElement(threeWord);
24)                                 result.addElement("nn");
25)
26)                                 i = i + 4;
27)                         }
28)
29)                         else
30)                         {
31)                                 if( wn.index.get( twoWord ) != null )
32)                                 {
33)                                         result.addElement(twoWord);
34)                                         result.addelement("nn");
35)
36)                                         i = i + 2;
37)                                 }
38)                                 else
39)                                 {
40)                                         result.addElement(tok1);
41)                                         result.addElement(tag1);
42)
44)                                 }
45)                         }
46)
47)                 }
48)         }
49)         return result;
50) }
```

Fig.24: The formulateConcepts( ) function.

Given the **qryVec** as input, we iterate through alternate elements in the vector (i = i+2 in line (6)), to pick up the first word and its tag (lines (9) and (13)). If the word is either a noun/adjective (line (15)), we form both three-token phrases, and two-token phrases with words that follow it (lines(18) and (19)). We then check to see if the three-token phrase exists as entry in Wordnet (line (21)), if it does, we add it to the result vector, and add the token "nn" as its tag (lines (23) and (24)), and jump iterations to the 6th element that

follows (i += 2 form line (6), and i = i+2 from line (26)). That is, we compound the three words into one three-token entry, and ignore the last two words in the token as possible individual/two-token entries. On the other hand, if we fail to find the three-token word, we apply the same procedure for two-token phrase (lines (10), and lines (31) to (37)). If we still fail to find such a two-token entry, we leave them as individual words (lines (40) and (41)). We finally return the result, a modified list that contains all the tokens compounded into maximum possible multi-tokens (i.e. three or less). It is important to note that, the order of checking first for three-token entries, and then checking for two-token entries is crucial as we try to capture the longest possible multi-token entries.

5.5 Semantic Analysis

5.5.1 The Structure of the Semantic Objects

As mentioned earlier, every syntactic object (LingObj) contains a corresponding semantic object. At this semantic level, the primary functionality involves: (1) the instantiation of primitive semantic objects for terminal symbols, and (2) the composition of semantic objects to form other semantic objects (new wholes contained by syntactic objects that represent non-terminal symbols). The important challenges in accomplishing these tasks include: (1) formulating an effective/efficient disambiguation process, and (2) describing an algebra that allows for the manipulation and composition of these semantic objects.

Much like the object-oriented design at the syntactic level where all syntactic objects extend LingObj, as shown in fig.19, every object at this semantic level is a sub-type of a meaning object. Illustrated in fig.25, is the object diagram for semantic objects used in Digital Agora.

As shown in fig.25, the MeaningObj, the super type of all other semantic classes is a generic class that implements all the utility functions that may be required by other semantic classes. The WnNode objects are primitive semantic objects that represent

individual word-senses from Wordnet, whereas the various concept objects are composites of other semantic objects composed according to the grammar.

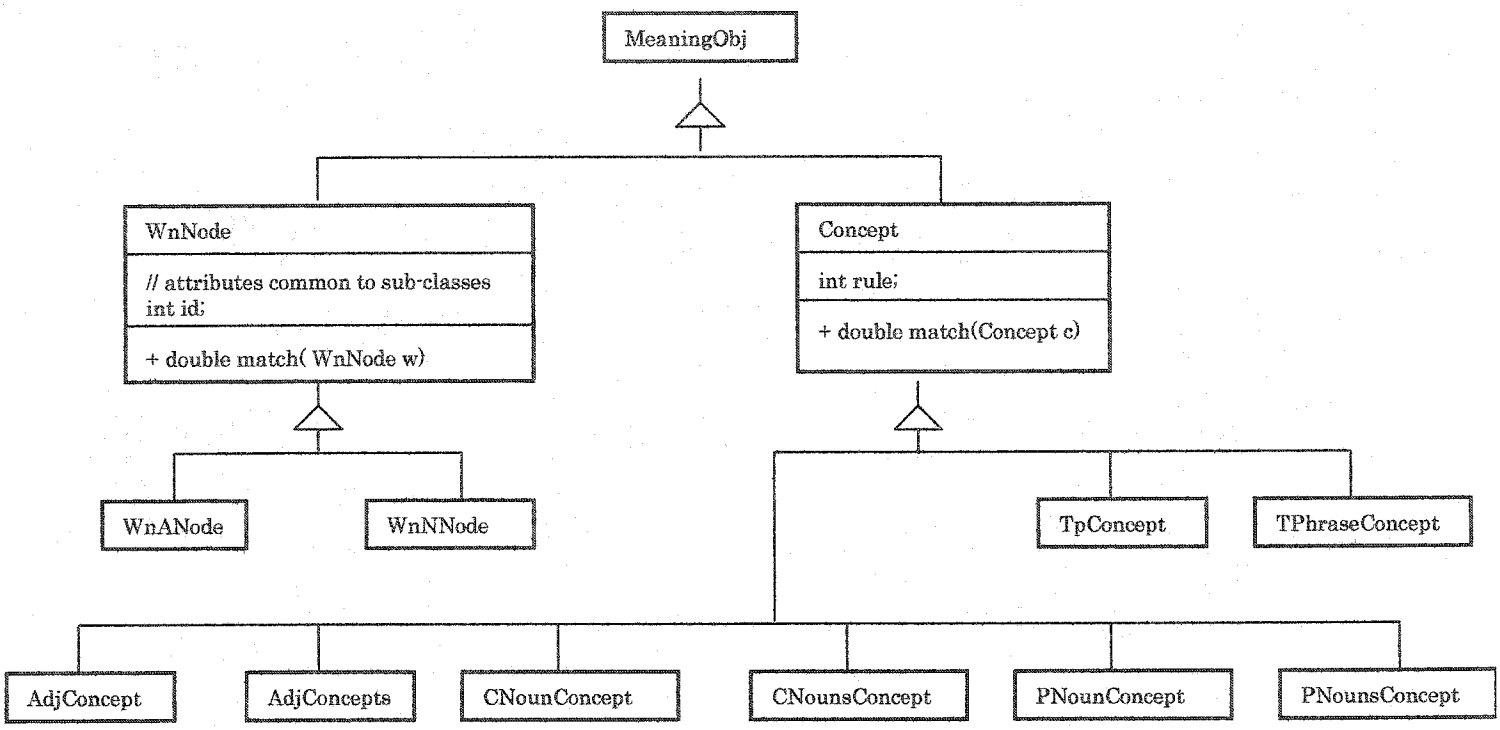


Fig.25: Object diagram of the semantic objects.

The Containment relationship between these various semantic objects is listed below:

Containment enforced by the lexical disambiguation.

- A CNounConcept contains one or more WnNNode objects.
- A PNounConcept contains one or more WnNNode objects.
- An AdjConcept contains one or more WnANode objects.

Containment enforced by the algebra deriving from the production rules in the CFG.

- A CNounsConcept contains one or more CNounConcept objects.
- A PNounsConcept contains one or more PNounConcept objects.
- An AdjsConcept contains one or more AdjConcept objects.
- A TpConcept conatins one or more CNounsConcepts/AdjsConcepts.
- A TPhraseConcept contains one or more TpConcepts.

5.5.2 The Primitive WnNode Objects

The WnNNode and the WnANode are the most primitive semantic classes, i.e. represent a particular noun-sense and adjective-sense respectively. These objects are contained by the CNounConcept/PNounConcept and AdjConcept objects (primitive concept objects) respectively. Each of these WnNode objects represent disambiguated senses of the concept objects they are contained by. We discuss the disambiguation process involved when instantiating these WnNode objects in section 5.5.4, where we discuss lexical disambiguation. To understand the data/functionality that these objects are responsible for, let us take a look at their various attributes and functions. Both these classes override the **match( )** function specified in the WnNode class. It is important to note that WnNode objects are matched only to other WnNode objects. Each of the subtypes of the WnNode class implement two variations of the **match( )** function (i.e. implement two match functions with different signatures). The idea is to implement not only matching noun-senses and adjective-senses respectively, but to also exploit polymorphism in inter-matching noun-senses with adjective-senses, and vice versa. The **relatedANS** attribute, a vector with noun-adjective relationships contained in Wordnet is the primary resource that drives this inter-type node match. The variations of the **match( )** function implemented by these classes are:-

```
public  double  match ( WnNNode  nounNode ){ }
public  double  match ( WnANode  adjectiveNode){ }
```

Each of WnNNode/WnANode classes have a set of attributes that represent both lexical/semantic information about the sense they represent (populated from the Wordnet semantic network). We refer the reader to section 3.4, for a detailed description of the various relations captured in wordnet. In this section, apart from discussing the meaning representation used to represent concepts in Wordnet, we study the various syntactic/semantic relations that are captured in the semantic network. The various immediate (vs computed) attributes of the WnNNode and the WnANode classes are listed below:

```
// WnNNode attributes

Integer     type;
int         polysymy;
Vector      synonyms;
Vector      hyponyms;
Vector      hypernyms;
Vector      isPartOf;
Vector      hasParts;
Vector      isMemberOf;
Vector      isSubstanceOf;
Vector      relatedANs;
Vector      antonyms;
Vector      relatedSyns;
Vector      similarSyns;
```

```
// WnANode attributes

Integer     type;
int         polysymy;
Vector      synonyms;
Vector      hyponyms;
Vector      hypernyms;
Vector      relatedANs;
```

Computed attributes, on the other hand, represent second, third and forth level relationships amongst these senses that are computed as a consequence of these immediate attributes. For example, **vehicle** is a hypernym of **car**, and **car** is a hypernym of **BMW**, **vehicle** is a second level hypernym of **BMW**, and so on. The strength of the relatedness between these various senses decreases with increasing (inversely proportional) distance in the semantic network. For example the strength of **car** is stronger than that of **vehicle**, when considering hypernyms of **BMW**, and so on. As space/time consuming as these additional attributes are, they are very important representations that capture distant relationships between various concepts in the semantic network. Some of the computed attributes in these objects include:

```
// computed attributes in WnNNode/WnANode

Hashtable   recursiveHyponyms;
Hashtable   recursiveHypernyms;
Hashtable   recursiveComponents;
Hashtable   recursiveCoordinates;
```

As a trade-off in quality for efficiency, we restricted the computed attributes to a maximum of 4 levels including the immediate ones. These attributes are stored in Hashtables (unlike vectors used for immediate attributes), to facilitate a handle on their levels of occurrence, based on which we compute their strengths. In brief, the sense number of an instance is the key, and their level of occurrence is the element.

Given WnNode objects populated with these attributes, matching two such nodes becomes simply a process of quantifying the similarity amongst them. Let us take a look at one of the **match( )** functions in the WnNNode class to get a better feel for how such similarity can be quantified:-

```
1)  public  double  match( WnNNode  nounNode )
2)  {
3)        double  conceptMatch = 0.0;
4)
5)        double   hypernymBias  =  0.75  *  isAHypernymOf( this, nounNode );
6)        double   hyponymBias  =  0.75 *isAhyponymOf(this, nounNode );
7)        double   isAPartOfBias  =  0.70  *  isAPartOf( this, nounNode);
8)        double   hasPartsBias  =  0.70  *  hasPartOf( this, nounNode );
9)        double   isMemberOfBias  =  0.70  * isAMemeberOf( this, nounNode );
10)       double   isSubstanceOfBias  =  0.70  *  isASubstanceOf( this, nounNode );
11)       double   isAntonymBias  =  0.70  *  isAnAntonymOf( this, nounNode );
12)
13)       onceptMatch  =  (hyponymBias + hypernymBias + isAPartOfBias + hasPartsBias +
14)                         isMemberOfBias + isSubstanceOfBias + isAntonymBias )  /  7;
15)
16)      return  conceptMatch;
17)  }
```

Fig.26: The match( ) function that matches WnNNodes.

As illustrated in fig.26, we call functions **isAHyponymOf( ), isAHypernymOf( )** (lines(5) to (12), and so on, to determine the relatedness of one concept to another. Apart from determining the existence of such relationships, each of these functions returns a double value between 0 and 1 that represents the strength of the relationship. For example, immediate relationships would have a strength of 1.0, and for every level further away we reduce 0.25. Therefore, the value returned by the function call to the **isAHypernymOf(WnNNode vehicle)** function, with the calling concept **BMW** would be 0.75. To capture the importance of certain relationships over others, we further modify these weights based on the relationship that they denote. For example, the hypernym/hyponym match is given more weight than the others, as shown in lines (5) and (6) (0.75 * isAHypernymOf(this, nounNode)). Lastly, we normalize the sum of all these matches to finally determine a node-to-node match (line (13)).

5.5.3 The Concept Objects

.

As with the case of WnNode objects, the concept objects also extend (are sub types of) the MeaningObj. The basic concept objects are the CNounConcept's, AdjConcept's and PNounConcept's that correspond to the terminal symbols in the CFG. Every CNounConcept contains one or more WnNNodes. This is so because, for most words, there is more than one sense that is selected during the disambiguation process (discussed in the section 5.5.4). This happens if more than one sense is relevant to a given context. Similarly, an AdjConcept may contain one or more WnANodes that represent the various disambiguated senses for an adjective.

All other concept objects are generated according to the production rules in the CFG. The CNounsConcept's contain one or more CNounConcept's, the PNounsConcept's contain one or more PNounConcept's, and the AdjsConcept's contain one or more AdjConcept's, in accordance to the following production rules:

```
CNouns  ::= CNoun  CNouns
          | CNoun

Adjs  ::= Adj  Adjs
        | Adj

PNouns  ::= PNouns  PNoun
          | PNoun
```

Similarly, the TpConcept's, and the TphraseConcept's are generated in correspondence to the following set of rules:

```
TPhrase  ::= Tp  Preposition  TPhrase
                | Tp

Tp  ::= Adjs  CNouns
      | Cnouns
      | Pnouns
```

As mentioned earlier, sub-types of a concept are populated in accordance to the syntactic objects instantiated during the parse. Therefore, as with the case of the parse, where we use the production rules of the CFG to compose syntactic objects to form larger syntactic objects, we require an algebra that corresponds to the CFG, to do the same with the concept objects. Two primary issues that are important at this concept level include: (1) the instantiation of these composite concept objects, and (2) the algebra used to match different concept objects.

The CNounConcept, PNounConcept, and the AdjConcept represent primitive concept objects that correspond to the terminal symbols, and are populated as soon as the parse identifies either a CNoun, PNoun or an Adj respectively, as a part of a Tp. The respective LingObj that will contain these semantic objects is responsible to instantiate them. To get a better feel for where these primitive concept objects might be instantiated in the process, take a look at line (16) in fig.20 (the definition of the CNoun syntactic object). The instantiation of these primitive concept objects involves disambiguating the given token that the concept represents, and in turn populating a set of relevant WnNode objects to represent each of the disambiguated senses (functionality carried out in the object constructors). Illustrated below is the definition of the CNounConcept:

```
1)  class CNounConcept extends Concept
2)  {
3)          Vector disambiguatedNodes = new Vector();
4)
5)          CNounConcept() {}
6)          CNounConcept(String word, Vector context)
7)          {
8)                  Vector disambiguatedSenses  =  disambiguate(word, context);
9)                  for( int I=0; I<disambiguatedSenses.size(); I++)
10)                 {
11)                         Long  oneSense  =  (Long) disambiguatedSenses.elementAt(i);
12)                         WnNNode  oneNode =  (WnNNode)  wordnet.get(oneSense);
13)                         DisambiguatedNodes.addElement(oneNode);
14)                 }
15)         }
16)
17)         // to be discussed later in this section
18)         public double match(CNounConcept  cn) { }
19) };
```

Fig.27: The definition of the CNounConcept class.

As illustrated in fig.27, the constructor of the CNounConcept class (line (6)), takes a word and the context as its argument. The word is then passed along with the context to the **disambiguate( )** function to receive a vector of disambiguated senses for that word (line(8)). The **disambiguate( )** function is discussed in greater detail in section 5.5.4, where we talk about lexical disambiguation. Once a list of **disambiguatedSenses** is returned by the **disambiguate( )** function, corresponding WnNNodes are retrieved from the loaded lexicon to populate the **disambiguatedNodes** attribute contained in the CNounConcept (lines (8) to (14)).

Unlike with primitive concepts, where they are instantiated directly from the linguistic objects, the composite concept objects that correspond to the non-terminal functions involve an additional step. An intermediate **apply( )** function is called to identify the correct constructor to be used when populating these concept objects, based on the parse rule that was successful. An instance of a Tp object calling such an **apply( )** function is illustrated in lines (22) through (25) of fig.21. The definition of this **TpApply( )** function is listed below:

```
1)  TpConcept  TpApply( int ruleNo, Vector rule )
2)  {
3)        TpConcept tpMeaning = new TpConcept();
4)
5)        if( ruleNo = = 1 )
6)        {
7)                  Vector cns = (Vector) rule.elementAt(0);
8)                  Vector adjs = (Vector) rule.elementAt(1);
9)                  tpMeaning = new TpConcept(cns, adjs);
10)       }
11)
12)       if( ruleNo = = 2 )
13)       {
14)                 Vector cns = (Vector) rule.elementAt(0);
15)                 tpMeaning = new TpConcept(cns);
16)       }
17)
18)       if( ruleNo = = 3 )
19)       {
20)                 Vector pns = (Vector) rule.elementAt(0);
21)                 tpMeaning = new TpConcept(pns);
22)       }
23)
24)       return tpMeaning;
25) }
```

Fig.28: The definition of the TpApply( ) function.

As illustrated in fig.28, the appropriate constructors have to be called when creating the TpConcept semantic object, as there are three possible ways to form a Tp (i.e. three production rules specified in CFG). Lines (5) to (9), lines (12) to (16), and lines (18) to (22) show the different scenarios where the three different constructors are called. The first scenario happens if the Tp was formed by the production rule **Tp : : Adjs Cns**, the second scenario happens if the Tp was formed by the production rule **Tp : : Cns**, and the third scenario happens if the Tp was formed by the production rule **Tp : : Pns**. Therefore, unlike with primitive concept objects like CNounConcept, and so on, the implementation of the composite concept objects like the TpConcept must facililate for the appropriate number of constructors with the relevant arguments. The definition of the TpConcept is illustrated below to better describe what we mean:

```
1)  class TpConcept extends Concept
2)  {
3)        AdjConcepts adjs = new AdjConcepts( );
4)        CNounsConcepts cns = new CNounsConcepts( );
5)        PNounsConcepts pns = new PnounsConcepts( );
6)
7)        TpConcept( ){ }
8)        TpConcept(Adjs adjs, Cns cns)
10)       {
11)               this.adjs = adjs.m;
12)               this.cns = cns.m;
13)       }
14)
15)       TpConcept(Cns cns)
16)       {
17)               this.cns = cns.m;
18)       }
19)
20)       TpConcept(Pns pns)
21)       {
22)               this.pns = pns.m;
23)       }
24)
25)       public double match(TpConcept tp) { // to be discussed later in this section }
26)  };
```

Fig.29: The definition of the TpConcept class.

As illustrated in fig.29, in correspondence to the **TpApply( )** function described in fig.28, there are three different constructors that determine three different ways to instantiate a TpConcept object (lines (8) to (13), lines (15) to (18), and lines (20) to (22)). Much like the TpConcept, other composite concept objects like the TPhraseConcept, and so on, also

implement multiple overridden constructors with different signatures[2] that correspond to the various production rules that apply to them.

Much like with the WnNode objects, the concept objects also have a **match( )** function that they implement. Every concept object implements a match that takes as its argument a concept of the same type. For example, line(18) in fig.27, and line (25) in fig.29 define the signatures of two such **match( )** functions. The list of such **match( )** functions implemented in their respective concept objects include:

- double match(TPhraseConcept oneConcept){ } //implemented in the TPhraseConcept class.
- double match(TpConcept oneConcept) { } //implemented in the TpConcept class.
- double match(PNounConcept oneConcept) { } //implemented in the PnounConcept class.
- double match(PNounsConcept oneConcept) { } //implemented in the PnounsConcept class.
- double match(CNounconcept oneConcept) { } //implemented in the CNounconcept class.
- double match(CNounsconcept oneConcept){ } //implemented in the CNounsConcept class.
- double match(AdjConcept oneconcept){ } //implemented in the AdjConcept class.
- double match(AdjsConcept oneConcept){ } //implemented in the AdjsConcept class.

The **match( )** in each of these cases is a function of applying the **match( )** of the respective components contained in each of these objects. In brief, the **match( )** of every concept object is ultimately broken down recursively to the most primitive **match( )** of their contained WnNode objects. OfCourse there are rules and heuristics implemented at every level to direct the build-up of these matches. It is these rules and heuristics that constitute the algebra of concepts. The **match( )** function in the primitive semantic objects is rather straight forward as it simply involves identifying the strongest match between the contained WnNode objects. The **match( )** function in the CNounConcept object is illustrated in fig.30:

As illustrated in fig.30, we iterate through all the WnNode objects contained in both the objects that need to be matched (lines (8) to (19)), and match every WnNNode from one object with every WnNNode from the other object to identify the maximum matching combination. This maximum match is then returned as the match between the two

[2] A signature is the set of arguments passed to a function.

CNounConcepts. If during the disambiguation process, the contained WnNodes in each of these concept objects are ordered according to their context-relatedness, this ordering can be incorporated in the CNounConcept **match( )** to further improve precision. Although we do not exploit this ordering in the current implementation of Digital Agora, we plan to incorporate it in the future.

```
1)  double match( CNounConcept cn )
2)  {
3)      double maxMatch = 0.0;
4)
5)      Vector cnNodes = cn.disambiguatedNodes;
6)
7)      // iterate through the WnNNodes contained in the calling object
8)      for( int i=0; i<disambiguatedNodes.size( ); i++ )
9)      {
10)             WnNNode oneNode = (WnNNode) disambiguatedSenses.elementAt(i);
11)
12)             // iterate through the WnNodes contained in the argument object.
13)             for( int j=0; j<cnNodes.size( ); j++ )
14)             {
15)                     WnNNode oneCnNode = (WnNNode) cnNodes.elementAt(j);
16)                     double oneMatch = oneNode.match( oneCnNode );
17)                     if( oneMatch > maxMatch )  maxMatch = oneMatch;
18)             }
19)     }
20)
21)     return maxMatch;
22) }
```

Fig.30: The match( ) function in CNounConcept class.

Unlike with the primitive concept objects, where we implement a rather straight forward algebra, the implementation of the **match( )** function in the other concept objects demands for a more complex one. In an effort to understand better the kind of algebra these matches entail, let us consider the match function implemented in the CNounsConcept class. The idea here is to match phrases like **information retrieval system**, and **document management engine**. When trying to match such concepts, it becomes necessary to bias the match based on the ordering of individual components in each of these concepts. Therefore, we use the location of each token in the phrase to reduce/increase the bias given to the match. For example, in this case more bias is given to matches between tokens that occur at the same position in each of these phrases, like **system/engine**, **retrieval/management**, and **information/document**, than the bias given to the **match( )** between concepts occurring at different positions like **information/**

management, and so on. At this level, such a algebra not only precisely captures the similarity between topics like **information retrieval system**, and **document management system**, but also succeeds in identifying that concepts like **company insurance**, and **insurance company** have a weak match. i.e. although both the topics **company insurance** and **insurance company** contain the exact same semantic components, the order of their occurrence reduces/increases their similarity. Although not explicit, this rule implicitly differentiates between head-nouns/modifier-nouns in each concept and accounts for them during the **match( )**. The **match( )** function implemented in the CNounsConcept object is illustrated below:

```
1)  public double  match( CNounsConcept  argConcept )
2)  {
3)        double   averageMatch  =0.0;
4)        Vector   thisNouns  =  this.cns;
5)        Vector   argNouns  =  argConcept.cns;
6)
7)        // iterate through the nouns in this object
8)        for( int i=0; i<thisNouns.size( ); i++ )
9)        {
10)                CNounConcept  oneThisNoun  =  (CNounConcept)  thisNouns.elementAt(i);
11)
12)                // iterate through the nouns in the argument object
13)                for( int j=0; j<argNouns.size( ); j++ )
14)                {
15)                        CNounConcept  oneArgNoun  =  (CNounConcept)  argNouns.elementAt(j);
16)                        double  oneMatch  =  oneThisNoun.match( oneArgNoun );
17)
18)                        if( i != j )  oneMatch  =  0.5 * oneMatch;   // bias the weight based on the positions.
19)                        averageMatch  =  averageMatch  +  oneMatch;
20)                }
21)       }
22)
23)       // normalize the match between 0 and 1.
24)       totalMatches  =  thisNouns.size( ) * argNouns.size( );
25)       averageMatch  =  averageMatch / totalMatches;
26)       return  totalMatches;
27) }
```

Fig.31: The match( ) function in CNounsConcept class.

As illustrated in fig.31, we iterate through all the CNounConcept's contained in the argument object (lines (13) to (20)), and through all the CNounConcept's contained in the node to which the **match( )** belongs (lines (8) to (21)), to match every CNounConcept from the former with every CNounConcept from the later (line (16)). As mentioned earlier, the bias of the match between CNounConcepts occurring at different positions in the vector is reduced by half (line (18)). The Vector implicitly contains the ordering of

the CNounConcepts in the CNounsConcept. The various computed matches are accumulated (line (19)) to finally be normalized between 0 and 1 (lines (24) and (25)).
In a similar fashion, such heuristics and rules drive the implementations of the **match( )** in the other concept classes. The difference however with TpConcept's, and TPhraseConcept's lies in that we ignore separators/prepositions from the semantic computation. Let us probe a little into this to understand why prepositions and separator classes were not implemented in the syntactic/semantic class hierarchies, although they are accounted for during the parse. Although we capture prepositions and separators when forming Tp's and TPhrase's during the parse, as observed in the object diagrams of the syntactic/semantic objects, we do not instantiate specific objects to represent them (i.e. we ignore their semantic contribution). This is primarily because the occurrence of prepositions/separators in TPhrases does not seem to change the subject-content underlying a phrase that contains them. Although they do alter the meaning of the phrase, and must be accounted for when performing complete NLU, ignoring them in Digital Agora seems to make negligible difference, as we are interested only in topic summaries (i.e subject-content underlying the phrases). For example, given the phrase **artificial intelligence in computer science**, we base all semantic computation involving this phrase purely on the meanings of **artificial**, **intelligent**, **computer**, and **science**, and ignore the contribution of the preposition **in**. The underlying assumption is that **in** here does not add any semantic value to the subject content of the phrase. Therefore, in an effort to simplify and speed-up the involved semantic computation, we ignore prepositions/separators, both at the syntactic and semantic levels. Although we ignore these prepositions and separators, the order in which the other components occur in these phrases is important and must be retained.

5.5.4 Lexical Disambiguation (Generating WnNode objects)

Semantic objects are populated on the return of the recursive-descent parse (i.e. when applying production rules to identify terminals and non-terminals). Lexical disambiguation is carried out at the level of the terminal symbols. As soon as the parse identifies a given token either as a common-noun, proper-noun, or an adjective, and

creates a syntactic object for the respective entity, a semantic counterpart for the syntactic object is also instantiated. As mentioned earlier, every CNounConcept, PNounConcept and AdjConcept, is composed of one or more WnNode objects. While forming a TPhrase, when we encounter a token that is either a noun, or an adjective, the correct sense of the word has to be disambiguated.

This disambiguation is based on a context window (a set of words occurring around the given word). In our approach, we ignore verbs and adjectives from the context, as there are no selectional-restrictions for adjectives/verbs available in the lexicon (Wordnet). Therefore, we simply use the nouns occurring around a given word as its context. After we conducted a number of tests using a variety of context-window sizes on a multitude of documents, we finally settled with using the entire document as the context-window. This seemed necessary to achieve the desired precision in disambiguation, primarily because, (1) we were ignoring the verbs/adjectives from the context, and (2) the Wordnet lexicon is incomplete and inconsistent in the few semantic relationships that it captures. Therefore, we needed as much information from the context as we could get. The underlying assumption in selecting the entire document as the context was that it is very unlikely that two different senses of the same word are used in the same document. In most cases, the important words (i.e. key-words that depict the core subject in the document) usually represent the same sense throughout the document. Therefore, during the pre-parse phase, immediately after tagging the document, we pick up all the nouns in the document to form the context set. Then WSD becomes a process of selecting sense(s) that have the maximum match with the context set. The **disambiguate( )** function is shown in fig.32.

As shown in fig.32, for every sense of the word to be disambiguated, every sense of a given context-word is matched (lines (14) to (37)). This is done because the context words themselves are not disambiguated. The maximum match is then selected with respect to that particular context-word. Similarly, such maximum matches are computed with respect to all the words in the context (lines (19) to (33)).

```
1)  public  Vector  disambiguate( String word, String posTag, Vector context)
2)  {
3)        WnNode  wordNode ;
4)        Vector disambiguatedNodes  =  new Vector( );
5)
6)        Vector wordSenses  =  new Vector( );
7)        Vector senseMatches  =  new Vector();
8)
9)        if( posTag.equals("ADJ") )
10)                wordSenes  =  (Vector)  adjex.get(word);
11)
12)       else wordSenses  =  (Vector)  index.get(word);
13)
14)       for( int  i=0; i<wordSenses.size( ); i++ )
15)       {
16)                WnNode  oneSenseNode  =  (WnNode)  wordSenses.elementAt(i);
17)                double  totalMatch  = 0.0;
18)
19)                for( int j=0; j<context.size( ); j+=2 )
20)                {
21)                        String contextWord  = (String)  context.elementAt(j);
22)                        Vector  contextWordSenses  =  (Vector) index.get(contextWord);
23)                        double maxMatch  = 0.0;
24)                        for( int k=0; k<contextWordSenses.size( ); k++ )
25)                        {
26)                                Long  oneContextSense  =  (Long) contextWordSenses.elementAt(k);
27)                                WnNNode  oneContextNode  =  new WnNNode(oneContextSense);
28)                                double oneMatch  =  oneSenseNode.match(oneContextNode);
29)                                if( maxMatch < oneMatch)  maxMatch  = oneMatch;
30)                        }
31)
32)                        totalMatch  =  totalMtach  +  maxMatch;
33)                }
34)
35)                double wordContextMatch  =  totalMatch / j;
36)                senseMatches .addElement( new Double(wordContextMatch));
37)       }
38)
39)       double  maxContextMatch  =  findMax( senseMatches );
40)       for(int i=0; i<senseMatches.size( ); i++)
41)       {
42)                double  oneSenseMatch  =  ( (Double)  senseMatches.elementAt(i) ).doubleValue( );
43)                if( maxContextMatch  -  oneSenseMatch  <=  0.20 )
44)                {
45)                        Long  oneDisAmbiguatedSense  =  (Long)  wordSenses.elementAt(i);
46)                        DisAmbiguatedSenses.addElement( oneDisAmbiguatedSense );
47)                }
48)       }
49)
50)       return  disAmbiguatedSenses;
51)  }
```

Fig.32: The disambiguate( ) function used in DigitalAgora.

Finally, to compute the match between the word-sense in concern and the entire context, the average of all these maximum matches is calculated (line (35)), where we total all the maximum matches, and divide this sum by the number of words occurring in the context. This now leaves us with a set of weights that determine the match of every word-sense to the entire context. As mentioned earlier, since we do not restrict the disambiguation to

precisely one sense, we iterate through the context-matches of all the senses (lines(40) to (48)), to capture all senses that differ less than 20% from the strongest match (i.e. chosen sense), (lines (43) to (47)). Although we enjoy a rather high precision in disambiguation using this approach as it is based purely on semantic/pragmatic computation, the process itself is very inefficient. This inefficiency results from the rather large computation that is necessary. To best understand the complexity involved, let us assume the example scenario where are trying to disambiguate the noun **line** (29 senses), against a context set with a 100 words. If we approximate an average of 5 senses for every word in the context, then, we have to perform 100 * 5 * 29 = 14,500 node-to-node matches just to disambiguate one word. Therefore, to disambiguate the entire document we would require atleast 14,50,000 such computational steps. That is if we assume that there are exactly the same number of words in document as the context. In an effort to compute an approximation of the complexity involved, let us assume without the loss of generality, that, all the words in the document are nouns. Then given such document with **m** words, and an average of **n** senses per word, the process incurs a complexity of $O(m^2n^2)$. To understand this complexity with respect to the **disambiguate( )** function illustrated in fig.32, the function incurs a cost of $O(m* n^2)$, (lines (14) to (37)), and since there are **m** such words in the document that have to be disambiguated, we have a total complexity of $O(m^2n^2)$. Although in the actual implementation, we further modified this algorithm by incorporating certain heuristics to reduce as much computation as possible, we were still left with a process that was highly inefficient. We stretched these heuristics to a little short of a point, beyond which any attempts to increase the efficiency would result in decline in the quality (precision). As will be discussed in the sections that follow, such inefficient disambiguation poses one of the major limitations that has to be resolved to make Digital Agora an efficient/efficient intelligent text-retrieval system.

5.6 The Topic Extraction Algorithm

The goal of the topic-extraction algorithm is to generate a subject summary of a given document based on key-topics that occur in it. The main focus here is to identify subject

themes that flow through the document. Apart from extracting the topics (TPhrases), the process involves ranking them according to their importance, and selecting a portion of these ranked topics to form the summary. Unlike traditional key-word based extraction, where a document is represented by key-words that occur the most, and concept-based extraction algorithms that represent the document by the most occurring key-concepts, in this algorithm we form a document summary that composes of the most important key-topics underlying the document. We use the word "important" here, as the selection of key-topics is not solely based on the number of times these topics occur in the document, but is also a measure of how and where they occur. Apart from using inbuilt topic-based extraction that caters to richer and more precise summaries than those of mere key-words/key-concepts, we further try to increase the quality of the summaries by determining the importance of these topics.

The importance of a topic is based on the consistency of its distribution throughout the document. Apart from the mere frequency of occurrence of the topic within the entire document, this algorithm also uses the frequency of topic occurrence within each paragraph, and the number of paragraphs that a topic occurs in, as measures when computing its importance. For instance, if the topic **artificial intelligence** has a document frequency of 7 (all occurrences in the same paragraph), and the topic **computer science** has a document frequency of 5 (occurring across three different paragraphs), then the topic **computer science** would be given more importance as it is distributed more consistently across the document. Unlike other approaches that use mere frequency of topic occurrence, this algorithm aims to identify topics that are continuous threads throughout the document. The underlying assumption here is that subjects that are referred to as continuous threads throughout the document, are most likely representations of what the document is about. The formula used to compute the importance of a topic in a document is illustrated below:

$$nwt(t, d) = \sum_{i=1}^{np(d)} \left[wt(i,t,d) - \left[\frac{wt(i,t,d) - bd(t,d)}{bd(t,d)} \right] \right]$$

where

- nwt(t,d) represents the normalized weight of topic 't' in document 'd'.
- wt(i,t,d) represents the weight (occurrences) of topic 't' in paragraph 'i' of document 'd'.
- np(d) represents the number of paragraphs in the document 'd'.
- bd(t,d) represents the best distribution of topic 't' in document 'd' (most ideal scenario).

bd(t,d), the best distribution of topic 't' in document 'd', is computed as follows

$$bd(t,d) = wt(t,d) / np(d)$$

where

- wt(t,d) represents the weight (simply no of occurrences) of topic 't' in document 'd'.
- np(d) represents the number of paragraphs in the document as discussed earlier.

Let us consider an example to better understand how this normalized weight of a topic in a document is computed. Consider a topic that has a weight of 10 (occurs 10 times), in a document with 5 paragraphs. Fig.33 illustrates five different scenarios of how this topic may be distributed across the document.

	scenario 1	**scenario 2**	**scenario 3**	**scenario 4**	**scenario 5**
paragraph 1	6	8	3	2	4
paragraph 2	2	0	2	2	3
paragraph 3	0	0	1	2	2
paragraph 4	0	0	2	2	1
paragraph 5	2	2	2	2	0
Nwt(t,d) (normalized weights)	**6**	**4**	**9**	**10**	7

Fig.33: Example scenarios that illustrate distributions of a topic in a given document.

The computation of nwt(t,d) as illustrated in fig.10, was computed by applying the formula based on the following values, np(d) = 5, and bd(t) = 10/5 = 2. Listed below are steps that illustrate how the formula was applied to compute the normalized weights nwt(t,d) for scenarios 1 through 5.

nwt(t, scenario 1)

→ (6 - ((| 6 – 2 |) / 2)) + (2 - ((| 2 – 2 |) / 2)) + (0 - ((| 0 – 2 |) / 2)) + (0 - ((| 0 – 2 |) / 2)) + (2 - ((| 2 – 2 |) / 2))

→ (6 – 2) + (2 – 0) + (0 – 1) + (0 – 1) + (2 – 0)

→ 6

nwt(t, scenario 2)

→ (8 - ((| 8 – 2 |) / 2)) + (0 - ((| 0 – 2 |) / 2)) + (0 - ((| 0 – 2 |) / 2)) + (0 - ((| 0 – 2 |) / 2)) + (2 - ((| 2 – 2 |) / 2))

→ (8 – 3) + (0 – 1) + (0 – 1) + (0 – 1) + (2 – 0)

→ 4

nwt(t, scenario 3)

→ (3 - ((| 3 – 2 |) / 2)) + (2 - ((| 2 – 2 |) / 2)) + (1 - ((| 1 – 2 |) / 2)) + (2 - ((| 2 – 2 |) / 2)) + (2 - ((| 2 – 2 |) / 2))

→ (3 – 0.5) + (2 – 0) + (1– 0.5) + (2 – 0) + (2 – 0)

→ 9

nwt(t, scenario 4)

→ (2 - ((| 2 – 2 |) / 2)) + (2 - ((| 2 – 2 |) / 2)) + (2 - ((| 2 – 2 |) / 2)) + (2 - ((| 2 – 2 |) / 2)) + (2 - ((| 2 – 2 |) / 2))

→ (2 – 0) + (2 – 0) + (2 – 0) + (2 – 0) + (2 – 0))

→ 10

nwt(t, scenario 5)

→ (4 - ((| 4 – 2 |) / 2)) + (3 - ((| 3 – 2 |) / 2)) + (2 - ((| 2 – 2 |) / 2)) + (1 - ((| 1 – 2 |) / 2)) + (0 - ((| 0 – 2 |) / 2))

→ (4 – 1) + (3 – 0.5) + (2 – 0) + (1 – 0.5) + (0 – 1))

→ 7

Clearly, as shown in fig.33, the best scenario is 4 where the distribution is consistent throughout the document. In scenario 4, not only do we find that the topic spans over the entire document (i.e. there is a continuous thread), but even the distribution across the document is consistent. Taking a second look at fig.33, we also observe that the next best scenario (3) is also one that illustrates such a continuous thread.

Although the extractor was implemented to form summaries based on topics that are well-distributed continuous threads throughout the document, it can be further enhanced by applying other linguistic heuristics like added bias to document topics, paragraph headings, and so on. Moreover, even normalization of topics within a paragraph will further improve the precision of the summary. Such normalization within the paragraph captures differences in paragraph sizes. For example a topic occurring in a paragraph of 20 phrases, should be given less importance than a topic occurring in a paragraph of 5 phrases, and so on. The focus of Digital Agora being that of subject-based text retrieval

(and not complete NLU), we stopped a little short of building such a complete topic-extractor by ignoring such enhancements, as we had discovered that the quality of the summaries without these modifications were precise enough to facilitate the desired subject-based searches. But we definitely, plan to revisit this area as a possible venue to further increase search once we are successful in improving the efficiency of the system.

Chapter 6

Towards an effective model for Lexical disambiguation: Initial Results

Most ambiguities in language still remain difficult unsolved problems in NLP/NLU. As important intermediate steps to most forms of language analysis and text processing, these unsolved ambiguities pose immediate challenges to the development of effective practical systems. As of late, stochastic (sub-symbolic) approaches to language processing have taken the forefront, primarily due to the lack of effective solutions from symbolic approaches. Although the accuracy of such statistical models has proven to be less than 90% with closed test sets, their performance further deteriorates drastically, when applied to open text. These approaches aim to resolve ambiguous situations in language through historic statistical data gathered by corpus analysis (Leacock et al., 1993), (Lehman, 1994) and (Mooney, 1996). Moreover, these approaches demand large sets of historic data under a multitude of pre-defined contexts. The infinitely many possible contexts in our everyday use of language make it a rather impossible task to build such a tagged corpus that is complete. Although from the perspective of a

computational linguist, one may argue in favor of such statistical approaches to NLP tasks, as immediately practical computational models, a theorist, supportive of symbolic approaches, would explain that there is a maximal bound to the effectiveness of such solutions, as they fail to advance our understanding of language.

Purely symbolic approaches, on the other hand have made limited progress in NLP. They have suffered primarily due to the lack of properly structured semantic networks that fail to commit to the correct set of ontological constraints in modeling the world around us. Semantic networks (general frameworks for representing knowledge), used in natural language understanding, are hierarchical representations of word meanings and conceptual associations between words (Allen, 1995). First introduced by Quillian in the mid eighties, these structures provide the basis for a symbolic approach to language analysis. Some of the more commonly known semantic networks include Wordnet, Cyc, Mikrokosmos, Pangloss and the EDR. The lack of formal heuristics, standards and principles in ontology design has led to the development of improperly structured taxonomies that underlie these semantic networks. Consequences of this malformation lead to (1) contradictive views of the world, (2) contradictions in the structure due to multiple inheritance and (3) the overloading of "is a" links (Guarino, 2000b), (Guarino, 1995), (Guarino, 1998) and (Stefik et al., 1993). Moreover, most of these semantic networks account only for a limited number of relations between concepts (like hypernomy, hyponymy, meronymy, entailment, and so on). In a symbolic approach to NLP/NLU, ambiguous situations are resolved by generating inferences to reason about semantic relatedness between the various concepts/conceptual-links in semantic networks. Therefore, the lack of sound and complete structures to represent semantic content poses a barrier to effective solutions in this direction.

The lack of well-founded computational models from either schools of thought, has driven a substantial portion of research in NLP to focus on various amalgamations of these two approaches to render more effective practical solutions. In this model, we propose one such attempt at lexical disambiguation. For the most part, we suggest parallel marker-propagation in semantic networks to disambiguate word senses. The

semantic network we take advantage of here is an extension to Wordnet. As is the common praxis when attempting pragmatic problems in AI, we use Wordnet as a starting point to approximate a commonsense linguistic-knowledge base on top of this dictionary. The focus here is (1) to illustrate how lexical disambiguation can be mapped to the problem of highly parallel searches in semantic networks, (2) to extend capabilities of traditional lexical disambiguation techniques to handle metaphoric use of language primitives and (3) to provide an account for each of the disambiguated senses by generating inference paths that drive the system into making these ambiguous choices.

Keeping in mind that this proposal is only a conceptual model, and that the upper-level extension to Wordnet that this model demands for, is only a simulation, we do not probe to attempt a certain sub-class of input scenarios that traditional selectional-restriction fails to cater to. Although we address these issues in this section, to show how this class is further reduced to a smaller subset, by providing for the metaphoric use of language primitives, we simply suggest the use of statistically pre-determined senses to tackle such scenarios. Well aware that statistically pre-determined senses is not the most optimal of solutions, we do not elaborate too much in this direction to remain un-deviated from the focus of the proposed model.

6.1 Selectional-restrictions in Lexical Disambiguation

Lexical ambiguity, like other ambiguities in English still remains a difficult unsolved problem. At this level, there are primarily two types of ambiguities, namely, syntactic ambiguity and semantic ambiguity. Syntactic ambiguity occurs when the same word belongs to multiple syntactic categories (for e.g. black is both a noun and an adjective). Disambiguation at this level focuses on identifying the category of a given word under a certain context. Semantic ambiguity on the other hand, occurs due to possible different meanings of same word (for e.g. **mouse** as in **rat** or **mouse** as in **hardware-device**). Disambiguating words at the semantic level deals with identifying the implied meaning of a given word under a certain context. The focus of this paper is on attempting a solution to semantic ambiguity, as solutions to syntactic ambiguity (part-of-speech

taggers and parsers), with high degrees of accuracy have become fairly standardized (Resnik, 1999).

Most words in English are highly polysemous, i.e. have multiple meanings (for eg. **star** has 7 meanings and **break** has 63 meanings). Listed below are the synsets (set of synonyms) and definitions for each of the 7 meanings of star as per Wordnet:-

1) **star:** (Astronomy) A celestial body of hot gases that radiates energy from thermonuclear reactions in the interior.
2) **star, ace, adept, sensation, maven, virtuoso, genius, hotshot, whiz, whizz, wizard, wiz:** Someone who is highly skilled.
3) *star: **Any celestial body visible (as a point of light) from the Earth at night.***
4) **star, principal, lead:** An actor who plays a principal role.
5) **star:** A plane figure with 5 or more points; often used as an emblem.
6) **star, headliner:** A performer who receives prominent billing.
7) **star, asterisk:** A star-shaped character * used in printing.

Moreover, words and meanings share a highly inter-tangled many-to-many relationship, where one word can have different meanings (commonly called senses of the word) and a meaning can be represented by many words (commonly called the synonyms of the word). Lexical disambiguation at the semantic level deals with isolating the appropriate sense of a given word under the specified context. The notion of disambiguating a word by looking at a small window around it (commonly known as the sliding window technique), was first suggested by Warren Weaver in 1955 (Jurafsky et al., 2000). Although first suggested in the context of machine translation, it is commonly used in lexical disambiguation. To appreciate the complexity of this process, let us re-visit the well-known examples:

8) The moon is a star that reflects light.
9) The astronaut is engaged to a star.

In sentence (8), the intended meaning of **star** under the context set **{moon, reflect, light}** is that of a celestial body as shown in sentence (1). In this example, disambiguation based on either statistical contextual occurrence or similarity measures between context words

is bound to isolate the correct sense of **star**, because **{moon, reflect and light}** accounts for a rather good context set that is representative of the intended sense. In sentence (9), however, this not the case, as the context around **star** is rather deceiving. The context set **{astronaut, engage}** tends to bias the decision towards that of sense (1), whereas the possible meanings are really senses (2), (4) and (6). These forms of sentences occur commonly in language, and therefore cannot be ignored. In such cases, thematic roles and selectional-restrictions prove to be competent solutions.

As mentioned earlier, semantic networks are classifications of concepts based on the inheritance ("isa") relationship between concepts. These semantic networks can be viewed as type hierarchies based on the subset relation, where every concept in the hierarchy can be treated as a set of all instances of that type, and sub-ordinate concepts can be treated as subsets of the super-ordinate set. This outlook towards semantic networks allows for various restrictions to be stated in terms of these sets in a rather intuitive way. For example, an adjective like heavy makes sense only if it is modifying a physical object like box, table, and so on. However, it does not make sense to talk about heavy events like seminar and debate. Therefore, if we treat the adjective heavy as a predicate of the form **heavy(x)**, then, we can constrain the argument of **heavy** to the set of physical objects (i.e. **physical_object** and its sub-types in the type hierarchy) (Allen, 1995). Such forms of constraints in the inheritance hierarchy are known as selectional-restrictions. A selectional constraint is a semantic constraint imposed by the lexeme on the concepts that can fill the various argument roles associated with it (Jurafsky et al., 2000). In a similar fashion, such forms of constraints can also be applied to verbs (both transitive and intransitive), and certain nouns as well.

10) eat(x).
11) drive(x,y).

In (10), plausible arguments for **eat** can be restricted to **food** and its subtypes. Similarly in (11), with the two-place verb **drive(x,y)**, the agent of the verb can be restricted to set of **persons**, and the object can be restricted to **vehicle** and its sub-types. Applying such a

restriction of the form **engage(person, person)** to sentence (9), makes it a rather straight forward process to identify that implied senses of **star** to be (2), (4) and (6) under the given context, either of which is a reasonable assumption. It is important to note that when we talk about **engage**, **person**, **drive**, **eat**, **food** and **vehicle** in these examples, we are specifically referring to a particular sense that each of these words represent, and not the lexemes themselves. Therefore, not only do these restrictions help disambiguate the arguments of the predicate (nouns), but they also aid in disambiguating the predicates themselves (i.e. verbs and adjectives). To illustrate what we mean, let us consider sentences (12) and (13).

12) $break_1$(John, can_1).
13) $break_2$(Sam, $rule_1$).

In sentence (12), given that the object of the sentence is a **can** (a physical object), we can identify the implied sense of the verb to be $\mathbf{break_1}$ as in the physical breaking into pieces. Similarly, in sentence (13), the object being a **rule**, we can identify the implied sense of the verb to be $\mathbf{break_2}$ as in a violation. In these examples, we use subscripts to denote differences in senses and assume without the loss of generality, that the sense we are talking about is the implied sense. As effective as they may seem with input scenarios (8) to (13), selectional-restrictions fail to accommodate for metaphorically used primitives in language. For example selectional-restrictions clearly do not apply to sentences like the ones illustrated below:

14) The astronaut is married to the stars.
15) The car drinks too much gas.

The context set **{astronaut, marry}**, around the word **star** in sentence (14), would clearly cause the selection to be biased to **star**, as in a person, when in truth the implied sense is **star**, as in a celestial body. Similarly, with (15), the entire sentence would be treated as implausible, because a **car** cannot **drink**. This discrepancy is primarily due to the lack of metaphoric meanings in existing knowledge-bases. Both these scenarios could have been tackled, if selectional-restrictions based on metaphoric senses of **marry** (as in

attachment) and **drink** (as in consume) of the form **marry$_2$(animal, thing)** and **drink$_2$(machine, substance)** are pre-defined. We illustrate later in this section how this class of problems can be overcome by integrating upper-level commonsense knowledge into existing semantic networks.

Yet another set of input scenarios that cannot be tackled via selectional-restrictions are those sentences where ambiguity remains, even after the application of these constraints. To illustrate what we mean, let us consider the example listed below.

16) The dentist painted the bridge.

In sentence (16), to disambiguate **bridge** purely based on selectional-restrictions deems worthless, as in this case both **bridge$_1$** (a construction that allows people to pass), and **bridge$_2$** (a denture anchored to teeth), will still remain ambiguous, because both these senses represent physical objects that can be painted. In such cases, semantic relatedness with other context words might prove relevant. For instance, in this example, semantic relatedness between **dentist** and **bridge** should steer the bias towards **bridge$_2$**. Although reasoning of this nature can handle such scenarios, we do not make any suggestions in this direction, primarily due to the immaturity of our research, and secondarily to retain the focus of this effort to that of mapping selectional-restriction to marker-propagation. In this model, we use readily available statistical data that define most commonly used senses in overcoming such scenarios, only for the sake of simplicity, completeness and ease of implementation in an effort to test our model.

Although the limited use of selectional-restriction in linguistic computational systems has proven very effective in disambiguating word senses, the following limitations pose a barrier to its practical applicability in real applications: (1) Its inability to address the metaphoric use of verbs, and adjectives in language, (2) the lack of implicit provision of selectional-constraints in existing semantic networks, and (3) the existence of a sub-class of input scenarios where ambiguity prevails even after the application of these restrictions.

Limitations (1) and (2) arise from the lack of a Formal Ontology that provides for ontological commitments of commonsense concepts relevant to language understanding. Although it is well accepted that large, scalable knowledge-bases of commonsense concepts are required to address such realistic pragmatic problems in various fields of AI, there seems to be no agreement on how to structure such repositories of commonsense knowledge. Therefore as is common praxis when attempting realistic problems in NLP, we use the Wordnet electronic dictionary as a starting point, to approximate a commonsense linguistic knowledge-base on top of this lexical dictionary. This upper-level extension is discovered as a consequence of various gramatic rules permissible on language primitives (Saba, 2001). Rather novel to ontology design, this approach as discussed in the later section, implicitly captures metaphoric-maps between various concepts and provides for selcetional-restrictions and thematic roles for verbs and adjectives within the structure. We aim to overcome limitations (1) and (2) by incorporating such an upper-level of commonsense knowledge on top of Wordnet and suggest the use of statistically pre-determined senses as an alternative to limitation (3).

6.2 The Upper-level Extension to Wordnet

It is by now well accepted that we require large stores of commonsense knowledge to perform pragmatic reasoning necessary to attempt ambiguous decisions in language. Although many efforts have been in continual progress over the last decade, there seems to be no consensus to an upper-level ontology of commonsense concepts. We will not be able to arrive at such a consensus unless this ontology is discovered, and not invented. (Mahesh et al., 1995), (Saba, 2001). Moreover, there seems to be little agreement on the methodology used to construct such structures, due to the lack of formal principles, criteria and standards in ontology design (Guarino, 1995), (Guarino, 1998), (Stefik et al., 1993). We approach building such an extension to Wordnet by using various cues in language (gramatic rules permissible on language primitives). In brief, we use pragmatically plausible thematic roles and selectional-restrictions to drive the classification of upper-level concepts. Apart from adding to Wordnet, missing metaphoric

sense definitions for language primitives, we use conventional image schemas as blue-prints to aid in identifying semantic maps between concepts (Saba, 2001).

To overview briefly the process in slightly more detail, we use verbs (transitive and intransitive) and adjectives to consistently classify nouns. We view concepts denoted by primitive lexemes as predicates with arguments. For now, we restrict these predicates to verbs and adjectives, although in future work we plan to extend them to action nouns as well. Then by constraining plausible arguments to a subset of the nouns in the lexicon, we isolate a restricted domain for each of these predicates. This predicate now becomes the criteria to split the lexicon. For instance, what kind of an object **C** is, can be determined from what verbs **V** and adjectives **A** can be applied as **V(C)**, **V(C, x)**, **V(x, C)** and **A(C)**. If we denote each predicate by a set of all concepts in its domain, then a subset operation across these predicates reveals a classification of nouns, with various verbs and adjectives acting as classification criteria at every split of the hierarchy. To illustrate what we mean, let us consider sentences (17) to (20):

17) Large Elephant (plausible)
18) Large Mountain (plausible)
19) Smart Elephant (plausible)
20) Smart Mountain (not plausible)

Sentences (17) and (18) are pragmatically correct because both **elephant** and **mountain** are plausible arguments for the predicate **large(x)**, similarly sentence (19) is also pragmatically correct because **elephant** is also a plausible argument for **smart(x)**. However, sentence (20) is pragmatically incorrect because **mountain** is not a plausible argument for **smart(x)**. This results in the set of **{elephant, mountain}** to represent **large(x)**, and the set of **{elephant}** to represent the predicate **smart(x)**. The application of the subset operation across these sets then results classification illustrated in fig.34.

Unlike adjectives and one place verbs (**walk(x)**, **eat(x)**, etc.), two place verbs like **feed(a, b)**, **touch(a, b)**, etc. carry a secondary role as explicit relations between concepts. That is, **touch(a, b)** would be a link that represents a relation between the concepts **animal** and

physical_thing, because the only plausible things that can touch are subtypes of **animal** and the only plausible things that can be physically touched (vs emotionally touched), are subtypes of **physical_thing**. Fig.35 illustrates this relation.

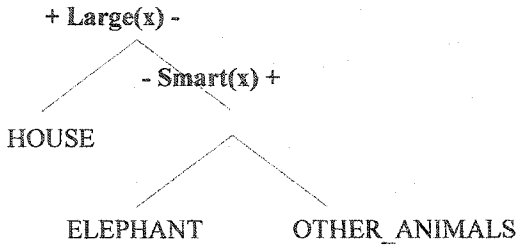


Fig.34: An illustration of adjectives as classification criteria.

The resulting taxonomy is a strict binary tree (with no multiple inheritance), where the nodes represent noun concepts and the edges represent the super/sub ordinate relationship between these concepts. Additional links that represent semantic relatedness between arguments of two place verbs are introduced on top of the taxonomy (Saba, 2001).

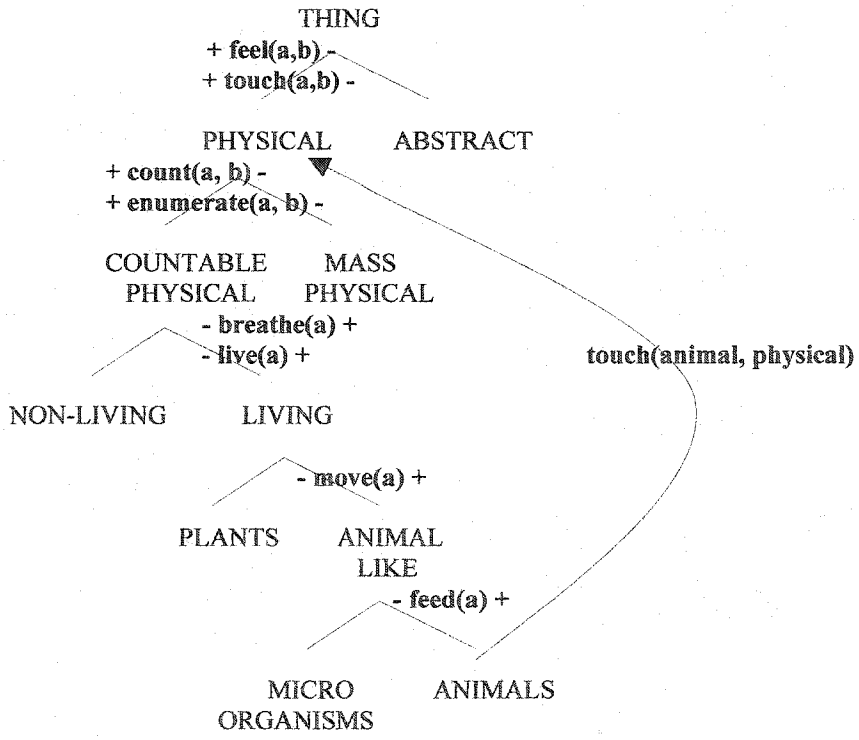


Fig.35: An illustration of relations represented by two-place verbs.

The system of conceptual metaphor underlying in language, i.e conventional mappings from one domain to another, is also incorporated into the structure through the use of kinesthetic image schemas (Saba, 2001). Generalizations governing polysemy, that is, the use of words with a number of related meanings is evidence for the existence of such a system of conventional conceptual metaphor (Lakoff, 1992). At the heart of the analogy of metaphor lies a structure-mapping process that is responsible for creating an isomorphic correspondence between semantic sub-structures that represent source and target domains. It is important to note that unlike its traditional use in knowledge representation, a domain here refers to a certain location in the structure and not to a specialized field. In a graph-based approach to meaning representation, isomorphic correspondence between semantic sub-structures is important because all meaning is expressed in terms of the structure, and isomorphism signifies structural similarity. Apart from acting as a substitute to a substantial portion of the multiple inheritance, this isomorphism in the structure also accounts for the representation of polysemy in language (Saba, 2001). To build such isomorphism into the structure, we use traditional kinesthetic image schemas suggested in metaphor research, as blueprints that aid in the construction of the taxonomy. Fig.36 illustrates how such isomorphism can be used to represent polysemy in language.

As shown in Fig.36, **run(x)** is a verb that can be applied to both **legged-animals** and **wheeled-vehicles**. However, the senses of **run(x)** applied to each of these concepts are different (different concepts represented by the same word). We use the verbs **run** and **fly** in example sentences to illustrate the difference in senses.

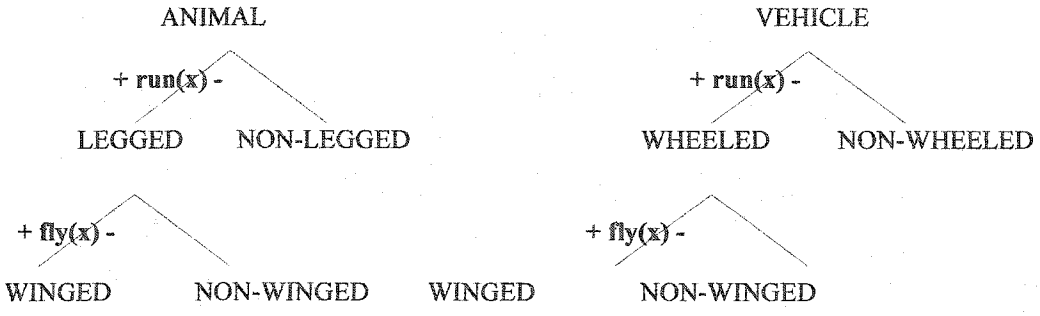


Fig.36: An illustration of isomorphism used in representing polysemy.

21) The dog **ran** very fast.
22) The bus **runs** between Toronto and Chicago.
23) The pigeon cannot **fly**.
24) Flight 376 **flies** out of the international terminal.

Although the example illustrated here shows how polysemy occurring with verbs is represented through isomorphism, it seems to be no different with the other categories of lexemes. In other words, polysemy that occurs with nouns and adjectives can also be represented in a similar fashion. The example illustrated in Fig.36 is a consequence of the so-called man-machine metaphor. More important than the specific metaphors themselves, are the kinesthetic schemas (illustrated in section 3.7), that provide general frameworks for such forms of mapping substructures. It is these schemas that dictate most of the isomorphism in the structure.

Although Fig.19 illustrates a scenario where both the source and the target domains are sub-structures on the physical side of the taxonomy, some of the strongest metaphoric maps hold between abstract and physical concepts (Saba, 2001), (Lakoff, 1987). In fact, most of the research in metaphors is concentrated around mapping abstract concepts to physical ones. For instance, water (physical) and information (abstract) have a very strong map. Anything we say about information, we can say about water.

Information leaks.	Water leaks.
Information flows.	Water flows.
A pool of information.	A pool of water.
We are flooded with information.	We are flooded with water.
We can filter information.	We can filter water.

On the other hand, it is only true that anything we say about information, we can say about water, whereas it is not necessarily true that anything we say about water can be said about information.

We drink water.	We cannot drink information.

Therefore, applying this criteria, the upper-level extension that we build on top of Wordnet, is a taxonomy of commonsense concepts which turns to be a strict binary tree, with isomorphic sub-structures that represent conventional metaphors underlying in language. The lack of formal definitions to differentiate commonsense vs domain-specific concepts, has driven us to assume the following heuristic: any concept that cannot be partitioned any further into a strict binary split based on verbs and adjectives is considered irrelevant at the commonsense level. Therefore, every strand of the binary tree is terminated at a point, where it cannot be split any further based on the suggested criteria. Rather interestingly, we find that starting by classifying mid-level concepts and then working around them, like in a jigsaw puzzle, turns out be more feasible than approaching the structure either bottom-up or top-down.

6.3 The Integrated Knowledge Base

Wordnet, an electronic dictionary based on semantic relations between word senses, is an effort to combine traditional lexicographic information and high speed computing. The construction of this semantic network, first started at the University of Princeton, under the supervision of Miller in the early 90's, is still under continual progress (Miller, 1998). Listed below are some of the key components of the integrated knowledge base (including traditional Wordnet 1.6 files).

A Gloss file: A file that maps every lexeme used in the definition of some concept, to all the concepts that use it in their definitions. An example entry from Wordnet's Gloss file is illustrated below:

25) airborne 1,00634314 1,03448058 1,05491755 2,01322256 3,00094341

This entry in the gloss file maps the word **airborne** to senses 0063414, 03448058 and 05491755 from the Noun.dat file, sense 01322256 from the Verb.dat file, and sense 00094341 from the Adj.dat (adjective data) file. It is important to note that these sense numbers not only provide a unique id for each of these concepts, but also represent the

offsets for each of these concepts in the respective data files. The numbers 1 to 4 occurring before each sense denote the Part-of-speech of the sense that follows (i.e. identifies the data file where the respective sense can be found. 1 denotes the Noun.dat file, 2 denotes the Verb.dat file, 3 denotes the Adj.dat file, and 4 denotes the Adv.dat file. Let us examine the definitions of each of these sentences to understand why the gloss of **airborne** points to them:

26) **00634314:** [air_reconnaissance]: reconnaissance either by visual observation from the air or through the use of **airborne** sensors.
27) **03448058:** [surface-to-air_missile, SAM]: a guided missile fired from shipboard against an **airborne** target .
28) **05491755:** [takeoff]: the initial ascent of an airplane as it becomes **airborne**.
29) **01322256:** [fly, wing]: travel through the air; be **airborne**; "Man cannot fly".
29) **0009434:** [surface-to-air] operating from or designed to be launched from the ground against an **airborne** target; "surface-to-air missiles".

As illustrated in (26) to (29), these senses form the gloss of **airborne,** as **airborne** occurs in the definitions of these senses. Under the notion that two senses must be related if they use the same word to define what they mean, traditional NLP systems have used this gloss information to define semantic relatedness between the various senses (both within and across various POS categories). We, on the other hand find this relation rather weak, and therefore use it only as a supplementary resource to improve our understanding of a sense when defining various selectional-contsraints for it. This relationship is rather weak primarily because it is:

Incomplete: Only one definition is listed vs listing all definitions that can be possibly used to define a sense. In other words, not all the words that can possibly be used to define senses are listed in their definitions.

Ambiguous: Although a lexeme is being used in these definitions, it is not exactly clear as to which sense of the word is being used in each case.

In the example listed above, we are fortunate that **airborne** is rather un-ambiguous as a noun (only one meaning). Therefore in the listed example, it is reasonable to assume a semantic relatedness amongst senses (26) to (29), as we can be sure that the same sense of **airborne** is being used in each of their definitions. However, with highly ambiguous

words like **machine** (represents 6 possible senses and is used in the definitions of 152 senses), it becomes a challenge to identify the correct sense of **machine** being used in each case, and thereby gives rise to the possibility of building semantic relatedness amongst irrelevant concepts.

Index files: A set of files for nouns, adjectives, verbs and adverbs respectively, that map every lexeme to a set of senses that it possibly can represent. The entry from the Noun.idx file that maps the noun **machine** to all its senses is illustrated below:

30) machine n 6 3 @ ~ %p 6 6 02949521 06137250 07385642 02383458 06137074 02950365

Apart from listing all the possible senses that a word can represent, the .idx files also provide for an alternate way to access sense numbers in the .dat files. Since the senses themselves are byte offsets that define the locations of these senses in the respective .dat file, it provides for retrieval by direct access using built-in library functions like **file.seek(long offset)** in Java, and **fseek(long offset)** in C/C++. Although these entries also provide for other information like occurrence of the word in various tagged corpuses and so on, the only information relevant to our model include:- (1) The third token (the number 6 in this example) that represents the polysymy of the word, (2) the senses themselves (the last 6 tokens in this example), (3) and the ordering of these last six senses. The ordering is important, as these senses are ordered based on the commonality of their use defined by statistical experience and other such heuristics. We suggest using this familiarity ordering when attempting to resolve the sub-class of input scenarios that selectional-restriction fails to cater to.

Data files: A set of files for noun-senses, adjective-senses, verb-senses and adverb-senses respectively, that map every sense to a set of synonyms, a definition, a noun type and sets of semantic relations. Some of the semantic relations defined in the data files include: (1) hyponyms, hypernyms, holonyms, coordinates and meronyms for noun-senses, (2) hyponyms, hypernyms, troponyms and entailments for verbs, (3) synonyms

and antonyms for adjectives/adverbs. Let us examine a noun entry from the Noun.dat file to understand this better:

31) 02481557 06 n 04 computer 0 data_processor 0 electronic_computer 0 information_processing_system 0 019 @ 02949521 n 0000 ~ 02186972 n 0000 %p 02357012 n 0000 %p 02405724 n 0000 %p 02413762 n 0000 %p 02432404 n 0000 %p 02482070 n 0000 %p 02482181 n 0000 %p 02546025 n 0000 ~ 02570061 n 0000 %p 02578948 n 0000 %p 02579698 n 0000 %p 02798795 n 0000 %p 02887166 n 0000 %p 03011151 n 0000 ~ 03042473 n 0000 ~ 03047816 n 0000 ~ 03089030 n 0000 ~ 03303483 n 0000 | a machine for performing calculations automatically.

Information in this data file is arranged based on the offset determined by the unique sense number itself (i.e. the sense number 02481557 also determines the byte offset location of this information in the data file). The second token in the entry, in this case the number 6 determines one of 28 Wordnet types that the concept belongs to. The number 6 refers to the Wordnet type **noun.artifact**. Apart from the definition that follows the token "|", other relations listed here can be summarized as follows:

Synonyms: [computer, data_processor, electronic_computer, information_processing_system]
Hypernyms: [02949521]
Hyponyms: [02186972, 02570061, 03042473, 03047816, 03089030, 03303483]
Meronyms: [02357012, 02405724, 02413762, 02432404, 02482070, 02482181, 02546025, 02578948, 02579698, 02798795, 02887166, 03011151]

Information relevant to us in generating the modified data file includes: (1) the synonyms, (2) the hyponyms, and (3) the definitions. The synonyms and definitions serve to identify a particular sense, whereas the hyponyms serve as the children nodes along which the markers will traverse.

Restriction files: A set of files for verb-senses and adjective-senses respectively, that maps every sense to an agent root and an object-root in the formal-ontology data file. In cases where the sense represents either a one-place verb or an adjective, the object-root is left empty. This file has to be hand-crafted as it requires human intervention in determining plausible arguments for the various verb and adjective predicates, based on the Formal Ontology. Apart from the decision-making process itself involved in building

this structure, the entries themselves are straight forward. Example entries from the restriction file are illustrated below:

32) 176662 990770 2859872
33) 73088 5531291

Sentence (32) represents the entry for the restrictions on a two-place verb-sense where 176662 (**break**$_1$) is the verb, 990770 (**chordate**$_1$,), is the root noun-sense for the sub-tree of concepts that represent plausible agents for this verb, and 2859872 (**instrumentality**$_1$), is the root noun-sense for the sub-tree of concepts that constitute plausible objects for the this verb. Sentence (33) on other hand, is the restriction for a one-place verb, where 73088 (**break**$_2$), is constrained to 2560468 (**outbreak**$_1$: as in a epidemic) and its sub-types.

The Formal-Ontology File: A file of upper-level noun-senses that maps every sense to two children (i.e. a representation of the binary tree discussed in earlier sections). However, for senses that represent leaf nodes in the binary tree, every sense is mapped to its corresponding location in the traditional Wordnet noun file. For the most part, this file is also hand-crafted as it is also based on the upper-level Formal Ontology. The semantic relation amongst concepts in this file is the discovered "is a" relationship from the Formal Ontology. Illustrated below is one such instance from the modified data file:

34) 00002086 00008864 00008019

As illustrated in sentence (34), 00002086 denotes **living_thing**$_1$, 00008864 denotes **plant**$_1$, and 00008019 denotes **animal**$_1$. This entry represents the binary split at the node **living_thing**$_1$, where **animal**$_1$ and **plant**$_1$ are the two children. The process of integrating leaf nodes of this upper-level structure to Wordnet's traditional noun-data file is also a very tedious process, as it involves taking care of compatibility issues like defining missing senses and eliminating redundant ones, to overcome existant "is a" overloading at the lower levels.

6.4 The Marker Propagation Algorithm

The theory of viewing memory search as activations spreading from two or more nodes in semantic networks, to find intersecting paths, is not new to language processing. As early as, in the late 60's spreading activation was a suggested as a general frame-work for reasoning with semantic networks (Quillian, 1966). Moreover, as of late, marker propagation (a specialized form of spreading activation), is rather commonly used as a parallel computing strategy to generate inference paths that denote semantic relatedness between concepts in a semantic network (Harabigau, 1999), (Harabigau, 1996), (Charniak, 1986). The lack of implicitly incorporated selectional-restrictions in existing semantic networks, had ruled out the possibility of applying parallel marker propagation at attempting lexical disambiguation. Traditional reasoning strategies that use slectional-restrictions are either sequential or incremental models. The upper-level extension to Wordnet proposed in this model, derived as a consequence of various gramatic cues permissible on language primitives, has embraced lexical disambiguation (via selectional-restriction) into a large class of unexplored problems that can be parallelized.

The underlying structure can be viewed as a graph with nodes, arbitrarily mapped onto a large number of processors, and a set of markers traversing the network based on some propagation rules. Markers are process threads that propagate from one node to another. Every marker keeps track of its starting node, and every other node that it passes through along its path. The rules of propagation implemented in this model are rather simple, in the sense that very little functionality is implemented into the marker. Moreover, the model is further simplified as the problem demands only for an asynchronous algorithm. Primarily so, because every path is independent of every other path and there is no need for inter-marker communication, except between parent and child, during instantiation.

Upon the arrival of a marker at a particular node, as many new markers as there are outgoing edges (children) from that particular node are spawned, in the form of a spread. Unlike traditional marker propagation in semantic networks, where the aim is to find intersecting paths that occur due to colliding markers, in this model, markers have pre-

defined destinations that they are seeking to find. The path of the marker that finds this pre-defined node is the result we are after.

In an effort to explain the process with respect to our problem at hand, we assume that the knowledge base is loaded, and that the input to the algorithm is the logical representation of a given English sentence. This is a reasonable assumption to make, as there are fairly standardized POS taggers and morphological analyzers with high degrees of accuracy.

35) A cat broke the mouse.

Therefore, without the loss of generality, we assume that sentence (31) is pre-processed to a predicate of the form **break(cat, cup)**, where break is tagged as the verb, and cat/mouse are tagged as nouns. The process we illustrate below aims to disambiguate the right senses for **break**, **cat** and **mouse**.

1) Retrieve a list of senses that the verb "break" possible represents (verb index file).
2) For every sense in this list:
   2.1) Identify the agent root and the object root (verb restriction file).
   2.2) Place a marker on the agent root and pass "cat" as its destination argument.
   2.3) Place a marker on the object root and pass "mouse" as its destination argument.
   2.4) Propagate all markers.

The process propagate involves the following steps:

3) If the destination argument is not a synonym of this node.
   3.1) For every child in the node
      3.1.1) If the node has children.
         3.1.1.1) Create a new marker.
         3.1.1.2) Pass the destination to the new marker.
         3.1.1.3) Add the name of the current node to the new marker's path.
         3.1.1.3) Propogate the new marker.

4) If the destination argument is a synonym of this node.
   4.1) Return the marker at that node as the path of the traversal.
   4.2) Return the node as the disambiguated sense.

In brief, the propagation starts at root nodes of the sub-tree of plausible agents/objects for every sense of **break**. Respective destinations are passed to each of these markers. A marker continues to propagate to the children of the residing node until it finds its predefined destination. Therefore, every marker continues to propagate either until its

strand is complete (i.e. the current node has no more children), or until its destination is found. The propagation step involves creating new markers, passing down the destination, updating their history (i.e. path), and in turn propagating them. This recursive process works in the form of a spread, until the desired sense (sometimes more than one sense) is found.

Fig.38 illustrates the selected paths that account for disambiguating the correct senses of **break**, **cat** and **mouse** for the input sentence **"a cat broke the mouse"**. The highlighted nodes represent selected paths that result in the respective disambiguated senses. As **break**, like many other words in English, is highly polysemous as a verb (63 senses), roughly 400 such paths have to be generated to make such an inference. This rather high degree of independent asynchronous processing that this model entails for, makes it necessary to view lexical disambiguation (via selectional-restriction) as a problem that demands parallel processing.

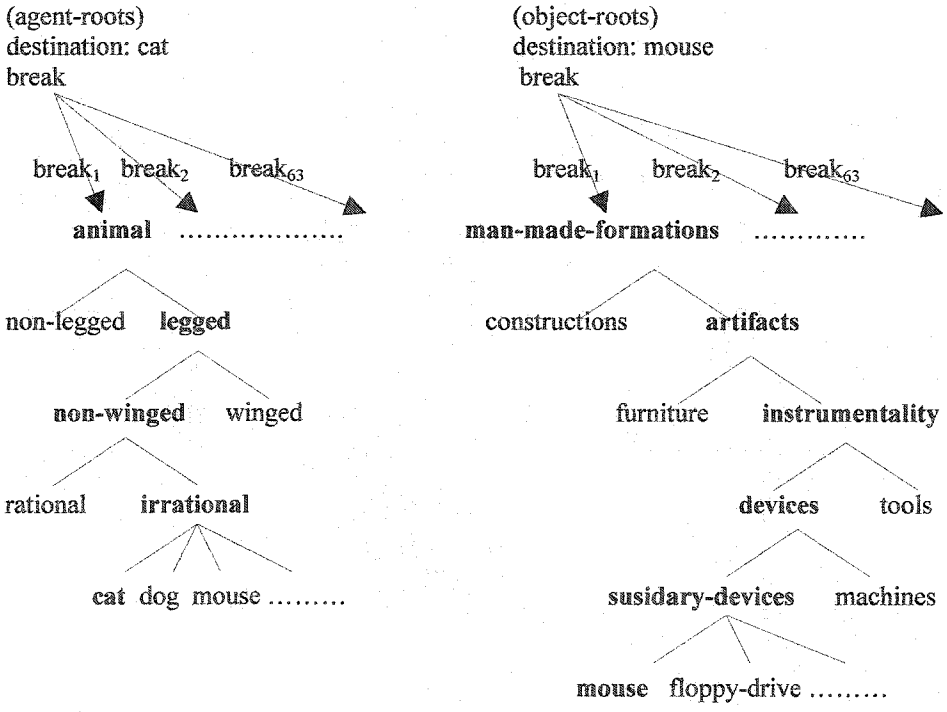


Fig 37: Illustrating selected marker paths in disambiguating "a cat broke the mouse".

In this model, we permit every thread to execute until completion, primarily because most times more than one sense is relevant to a certain context. As with the class of exceptional input-scenarios that selectional-restriction fails to capture, more than one sense is selected, in which case, the selected senses have to be further pruned. Although in this model, for the sake of completeness, we suggest using the familiarity ordering suggested by Wordnet to make decisions in such scenarios, we admit that this is not the most optimal solution, and the use of semantic distance between context words or other statistically tagged corpuses may prove more accurate.

```
1)  public void run() {
2)      Vector syns = (Vector) ontologySyns.get(startPoint);
3)      If ( syns.contains(destination) ) {
4)          If ( type.equals("agent-marker") ) {
5)              If ( !disambiguatedAgents.containsKey(verbSense) ) {
6)                  disambiguatedAgents.put(verbSense, startPoint);
7)                  pathHistory.put(startPoint, history);
8)              }
9)          }
10)         else if ( !disambiguatedObjects.containsKey(verbSense) ) {
11)             disambiguatedObjects.put(verbSense, startPoint);
12)             pathHistory.put(startPoint, history);
13)         }
14)     }
15)     else {
16)         Vector children = (Vector) ontologyHyps.get(startPoint);
17)         for(int i=0; i<children.size(); i++) {
18)             Long oneChild = (Long) children.elementAt(i);
19)             Vector newHistory = (Vector) history.clone();
20)             newHistory.addElement(oneChild);
21)             Marker m = new Marker(id++, oneChild, newHistory, destination, verbSense, type);
22)             m.run();
23)         }
24)     }
25) }
```

Fig 38: The Run( ) method of the Marker Class.

The parallel disambiguation algorithm was implemented using Java, and exploits Java's multi-threading environment to achieve the necessary parallelism. Java threads are light-weight processes that are capable of executing portions of a program concurrently. Therefore, every marker propagated across the network is mapped to a unique thread of execution, and is thereby allowed to propagate concurrently with other markers. To ensure complete independency during execution, all functionality that controls the propagation (i.e. propagation rules) is built into the markers themselves. Thereby, apart from the integration of the end results of each of these threads, this embarrassingly-

parallel computational model eradicates the need for inter-thread communication, during execution. The run() method of the Marker class is illustrated in Fig.39.

The Marker class extends class Thread form the Java API, to inherit all inbuilt generic capabilities necessary for concurrent processing. Like any extension of the Thread class, the run method is overridden to specify functionality of the marker during its life-time. Every Marker object has the following attributes:

- **an id:** An integer that uniquely identifies the marker.
- **a start node:** The sense-no of the noun node where the marker originates from.
- **a history:** A vector that represents the path traversed by its parent.
- **a destination:** The word it is trying to disambiguate.
- **a verb-sense:** The sense-no of the verb for which it was triggered
- **a type:** A String that distinguishes between agent-markers and object-markers.

All these attributes are populated during the instantiation of a marker (i.e. are passed as arguments to the constructor of the marker). Given these attributes, the marker performs one of two series of steps, based on whether or not its current node is the destination it is looking for. This is determined by checking whether or not the set of synonyms of the current node contains the destination string (line 3). As illustrated between lines (3) to (13), if the current node is the destination it is looking for, the marker adds the sense to a shared list of possible candidate senses in disambiguating the noun. Moreover, the noun (agent/object) being disambiguated is determined by the type of the marker. If the marker is an agent-marker, the node is added to the **disambiguatedAgents** list (line 6), and if the marker is an object-marker, the node is added to the **disambiguatedObjects** list (line 11).

On the other hand, if the current node proves not to be the destination that the marker is looking for, the marker simply propagates new markers to all outgoing children (lines 15 to 24). Apart from the **verb-sense**, and the **destination**, all other parent attributes are modified when creating new markers. A new **id** is given to every child-marker by incrementing the static variable **id** (line 21), the **startPoint** for each of these new markers is modified (oneChild argument in line 21), and the **history** vector is updated by adding the child to the path of the new marker (line 20).

This recursive instantiation of new markers continues concurrently with other executing threads, until either the individual destination is found or the strand is complete (i.e. current node has no more outgoing children). As illustrated earlier, all paths of successful markers are stored in separate lists for agents/objects respectively. Upon termination of all threads, **disambiguatedAgents** constitutes a list of various successful verb-senses connected to possible agent-senses, and **disambiguatedObjects** represents a list of various successful verb-senses connected to possible object-senses. Both **disambiguatedAgents** and **disambiguatedObjects** are persisted using hashtables, where the verb-sense is the key and the respective noun-sense is the stored element. Therefore, a simple selection of common keys amongst these two hashtables, reveals a disambiguation for the sentence (i.e. a predicate of the $\mathbf{verb_1(agent_1, object_1)}$). Although we have only described the disambiguation process for two-place verbs, with one-place verbs this process is further simplified, whereby there is no need for object-markers and **disambiguatedObjects**. Then, **disambiguatedAgents** would directly represent the possible disambiguations of the sentence (i.e. a predicate of the form $\mathbf{verb_1(agent_1)}$).

6.5 An Example Execution

To measure the quality and the speed-up of the ambiguity-resolution process, the marker-propagation algorithm was tested to disambiguate the input **"a mouse broke a mouse"**. Due to the lack of a complete Formal Ontology of upper-level concepts, a simulation of such a knowledge-base was created around the verb **break**. The various senses of the verb **break** (Wordnet 1.6), were inserted into appropriate levels of the Wordnet's semantic network of noun concepts. A mini knowledge-base of 13,560 concepts that deemed relevant to the selectional-restrictions of these various senses was isolated. The primary intention behind isolating this mini-knowledge base was to reduce the size of the knowledge-base to avoid added file seek time during the execution of the algorithm, by pre-loading all necessary data into memory before executing any test scenarios.

Although this simulated knowledge-base has inconsistencies ("is a" overloading), traditionally found in Wordnet, it provides sufficient ground to test the recall, precision and speed-up of the ambiguity-resolution process. The steps involved in constructing this simulated knowledge base for these tests include the following: (1) hand-crafting the restriction file for the various senses of break, by determining agents-roots and object-roots from the traditional Wordnet Noun.dat file, and (2) isolating the necessary concepts that occur in the sub-trees of which these agents/objects constitute the root.

The mini knowledge base was isolated by tracing the hyponyms below these root nodes recursively. The recursive function used to isolate these concepts is illustrated in Fig 40.

```
1)  private void recHyps(Long oneLong)
2)  {
3)      long offset = oneLong.longValue();
4)      try {
5)          nounDataFile.seek(offset);
6)          String input = nounDataFile.readLine();
7)          miniKnowldgeBase.writeUTF(input);
8)          if( input.indexOf("~") == -1 )
9)            return;
10)
11)         StringTokenizer tokenizer = new StringTokenizer(input, " ");
12)         while( tokenizer.hasMoreTokens() )
13)         {
14)           String tok = tokenizer.nextToken();
15)           if(tok.equals("~"))
16)           {
17)               Long hyp = new Long(tokenizer.nextToken());
18)               recHyps(hyp);
19)           }
20)       }
21)     }catch(IOException io){
22)       System.out.println(io.toString());
23)     }
24)  }
```

Fig 39: The recHyps( ) method, to isolate the mini-knowledge base.

The recHyps() function takes a long offset value as its argument, as illustrated in line (1). It reads the data at the given offset from Wordnet's noun data file (line 5), and writes it to the mini-knowledge base (line 6). As long as this sense has hyponyms (i.e. children), the recHyps() is called recursively for each of the children (line 18). This recursive process terminates only if the given sense does not have any outgoing children (lines 8 and 9). The "~" is a delimiter used in the Wordnet Noun.dat file to represent hyponym entries.

As this simulated knowledge-base is not a strict binary tree, and has multiple-inheritance, without the loss of generality, we conclude that both the quality and efficiency of the algorithm will be significantly better, when applied to the real Formal-Ontology, as its underlying taxonomy is a strict binary tree with no inconsistencies (Saba, 2001).

Passing the loaded knowledge-base and the predicate **break(mouse, mouse)** as input arguments, the program was executed to observe the following results.

- Total no of markers created: 40,019.
- Successful agent-markers (verb-agent combinations) of the form verb(agent, object).

    1) 1728889(1832174, mouse)
    2) 1681774(1832174, mouse)
    3) 0231588(1832174, mouse)
    4) 0176662(1832174, mouse)
    5) 0938146(1832174, mouse)

- Successful object-markers (verb-object combinations) of the form verb(agent, object).

    1) 1681774(mouse, 3020109)
    2) 0231588(mouse, 3020109)
    3) 0176662(mouse, 3020109)
    4) 0938146(mouse, 3020109)

- Disambiguated senses for break (i.e. intersecting verb senses from successful agent/object markers).

    1) 1681774 : [break] : divide into pieces, as by bending or cutting.
    2) 231588 : [break] : destroy the integrity of; usually by force.
    3) 176662 : [break] : render inoperable or ineffective.
    4) 938146 : [break, bust] : ruin completely; "He busted my radio!".

- Disambiguated senses for the agent mouse.

    1) 1832174 : [mouse] : any of numerous small rodents typically resembling rats.

- Disambiguated senses for the object mouse.

    1) 3020109 : [mouse] : An data input device that moves a cursor on a computer screen.

- Path traversed by marker to reach mouse (sense: 1832174).

---> [chordate]
---> [vertebrate, craniate]
---> [mammal]
---> [placental, placental_mammal, eutherian, eutherian_mammal]
---> [rodent, gnawer, gnawing_animal]
---> [mouse]

- Path traversed by marker to reach mouse (sense: 3020109).

---> [instrumentality, instrumentation]
---> [device]
---> [electronic_device]
---> [mouse]

Although the disambiguated senses are intuitively correct, we define precision, recall and speed-up to formalize the observed results in terms of quality and efficiency (Harabagiu, 1999), (Jurafsky, 2000).

1) **Precision:** ratio between the number of correctly disambiguated senses found by the system over the total number senses found by the system.

break = 4/4 = 1
$mouse_1$ = 1/1 = 1
$mouse_2$ = 1/1 = 1

2) **Recall:** ratio between the number of correctly disambiguated senses by the system over the number of correct senses in the knowledge base.

break = 4/4 = 1
$mouse_1$ = 1/1 = 1
$mouse_2$ = 1/1 = 1

3) **Speed-up:** ratio between the total number of marker propagations (sequential time), over the number of propagations along the longest path (parallel time). Without the loss of generality, we can choose 20 to be the length of the longest path because no strand in Wordnet extends over 20 levels.

speed-up = 40,019 / 20 = 2000.95

Unlike traditional IR systems, where precision and recall work against each other, in this model we observe that both these measures are high and as they are dependant on the consistent classification of the upper-level Formal-Ontology. Moreover, unlike traditional iterative/incremental computational models that implement selectional-restrictions, we see that speed-up is very high due to the embedded parallelism in this computational model.

6.6 Other Test Results

1) Disambiguating the predicate **break(human, table)**.

- Total no of markers created: 40630.

- Successful agent-markers (verb-agent combinations) of the form verb(agent, object).

1) 501113(1967203, table)
2) 176390(1967203, table)
3) 73223(1967203, table)
4) 749317(1967203, table)
5) 1829329(1967203, table)
6) 1109336(1967203, table)
7) 1728889(1967203, table)
8) 1109521(1967203, table)
9) 176662(1967203, table)
10) 501239(1967203, table)
11) 938146(1967203, table)
12) 1820618(1967203, table)
13) 270556(1967203, table)
14) 1681774(1967203, table)
15) 1418950(1967203, table)
16) 231588(1967203, table)
17) 1688962(1967203, table)
18) 529048(1967203, table)
19) 253202(1967203, table)
20) 1752871(1967203, table)

- Successful object-markers (verb-object combinations) of the form verb(agent, object).

1) 1681774(human, 3461269)
2) 231588(human, 3461269)
3) 176662(human, 3461269)
4) 938146(human, 3461269)

- Disambiguated senses for break (i.e. intersecting verb senses from successful agent/object markers).

    1) 176662 : [break] : render inoperable or ineffective.
    2) 938146 : [break, bust] : ruin completely; "He busted my radio!".
    3) 1681774 : [break] : divide into pieces, as by bending or cutting.
    4) 231588 : [break] : destroy the integrity of; usually by force.

- Disambiguated senses for the agent human.

    1) 1967203 : [homo, man, human_being, human] : any living or extinct member of the family Hominidae.

- Disambiguated senses for the object table.

    1) 3461269 : [table] : a piece of furniture having a smooth flat top supported by one or more vertical legs.

- Path traversed by marker to reach human (sense: 1967203).

---> [chordate]
  ---> [vertebrate, craniate]
    ---> [mammal]
      ---> [placental, placental_mammal, eutherian, eutherian_mammal]
        ---> [primate]
          ---> [hominid]
            ---> [homo, man, human_being, human]

- Path traversed by marker to reach table (sense: 3461269).

---> [instrumentality, instrumentation]
  ---> [furnishings]
    ---> [furniture, piece_of_furniture, article_of_furniture]
      ---> [table]

2) Disambiguating the predicate **break(news)**.

- Total no of markers created: 10985.

- Successful agent-markers (verb-agent combinations) of the form verb(agent).

    1) 633716(4982831)

- Disambiguated senses for break

    1) 633716 : [break, get_out, get_around] : be released or become known; of news; "News of her death broke in the morning".

- Disambiguated senses for the agent news.

    1) 4982831 : [news, intelligence, tidings, word] : new information about specific and timely events; "they awaited news of the outcome".

- Path traversed by marker to reach news (sense: 4982831).

    ---> [news, intelligence, tidings, word]

Chapter 7

Conclusion

Lexical disambiguation still remains a difficult unsolved problem in language. In this thesis we proposed a parallel computational model to attempt this problem. This method uses a marker-propagation algorithm to perform word sense disambiguation based on the relatedness of concepts in a semantic network. Unlike traditional methods, the suggested approach provides an account for each of the selected senses in the form of paths traversed by successful markers. Apart from discussing an extension to the Wordnet semantic network based on the Formal Ontology (a structure assumed by the algorithm), this report described the marker-propagation algorithm and illustrated test results when applied to an input scenario.

7.1 Summary

Identifying the need for language understanding to build intelligent software, we began this report by discussing various ambiguities in language that have to be resolved. As a

continuous theme throughout the report, we emphasized the need for large, scalable commonsense knowledge-bases to address realistic pragmatic problems to facilitate language understanding. Keeping in mind that most people communicate using language without knowing everything about any specialized domain, we differentiated commonsense knowledge from domain-specific knowledge to determine that portion of knowledge necessary to understand language. We probed into semantic networks as suitable structures to represent such forms of knowledge, and surveyed their various limitations in facilitating commonsense reasoning. Dating back to the early history of semantic networks, we discussed work done by pioneers in the field like Poriphery, Pannini and Quillian. To get a feel for how natural language systems reason with these structures, we studied default reasoning, spreading activation and selectional-restrictions as prominent techniques. Analysing the build-up of commonly used knowledge-bases like Cyc, Mikrokosmos, Wordnet and the EDR, we identified multiple-inheritance as a common ill-effect amongst all of them. Identifying that multiple-inheritance in these structures causes contradictive inferences, we proposed constructing an upper-level commonsense ontology for NLU tasks based on cues in language. As a solution to representing and reasoning with commonsense knowledge, we proposed the use of a Formal Ontology that: (1) has no multiple-inheritance, (2) incorporates metaphoric definitions of various senses, (3) suggests implicit classifications for verbs and adjectives, (4) implicitly accounts for class exceptions, and (5) distinguishes properly between roles and types.

After a thorough understanding of the various ambiguities in language, and the complexities involved in attempting solutions to resolve these ambiguities, we discussed Digital Agora: An experiment in intelligent text processing. Apart from showing in this experiment that complete NLU is not necessary for intelligent text processing, we demonstrated how topic-based text retrieval increases the precision quality of searches in open-ended corpus. The model and design of Digital Agora were analyzed to illustrate the various heuristics and trade-off's implemented to find optimal balances between performance in terms of speed and quality. Understanding the involved semantic/pragmatic computation to facilitate such intelligent topic-based text retrieval,

we identified the high complexity caused due to lexical disambiguation as the major drawback that hampered the completion of the system. We then proposed a parallel computational model that uses marker-propagation to address lexical disambiguation. Identifying that selectional-restrictions prove competent solutions in disambiguating word senses, we accounted their limited use to the lack of knowledge-bases that provide for implicitly embedded selectional-constraints in their respective semantic structures. Unlike traditional iterative or incremental computational models that used limited forms of selectional restrictions, given a semantic network with such embedded constraints, we discussed how lexical disambiguation can be mapped to the problem of an embarrassingly parallel search in semantic networks. Apart from proposing such a computational model that assumed a knowledge-base of upper-level concepts based on the Formal Ontology, we implemented and tested a parallel marker-propagation algorithm to disambiguate word senses. Due to the lack of a completed upper-level extension that this model assumes, a mini knowledge-base (a simulation of the Formal Ontology) was approximated around the 63 senses of **break** by isolating 13,560 concepts from the Wordnet electronic dictionary, to test the precision, recall and speed-up of the marker-propagation algorithm.

7.2 Future Work

As mentioned throughout this thesis report, the construction of the formal-ontology is a long-term process that requires an enormous number of man-hours. Moreover, as the formal ontology is one that must be discovered (and not invented), every step of its construction is open to philosophical debate. As much as we understand that a complete formal ontology that commits to all the ontological constraints used in language is necessary before we can have complete NLU, we do realize a partially complete ontology is sufficient for immediate applications in intelligent text-processing. Since it seems that selectional-restrictions provide a good methodology to discover this ontology, and efficient/effective lexical disambiguation is possible given such selectional-restrictions in the knowledge representation, continuing to build the language-based formal-ontology

remains our top-priority. Upon the completion of this partial ontology where we are at least able to build into the structure selectional-restrictions for verbs/adjectives, we will revisit Digital Agora with more rigor to implement the following enhancements:

- Incorporate verbs into the context when performing word sense disambiguation.
- Reduce the size of the window size to a maximum of 20 words (10 on each side).
- Experiment with and enhance the algebra of concepts that is currently being used.
- Experiment with and enhance the topic-extraction algorithm with other linguistic heuristics.
- Modify the underlying model of Digital Agora to facilitate the marker-propagation in lexical disambiguation.

As simple as making sense of utterances in language may seem to us as humans, so difficult is the problem when mapped to computers. Although continual efforts have been in progress since the advent of the computer to embed this capability into these machines, it still seems a long way to go before we can build a program that can understand language. Although many believe that this may never be possible, we disciples of Turing are not willing to accept defeat. The rest remains for time to tell.

Abernethy, F. N., Wu, J., Hewett, M. and Altman, B. R. (1999). SOPHIA: A flexible, web-based knowledge server. *IEEE Intelligent Systems, Vol.14, No.4.*

Allen, J. (1995). Natural language understanding. *The Benjamin/Cummings Publishing Company, Inc.*

Allen, J. (1984). Towards a general theory of action and time. *Artificial Intelligence, 23(2), pp. 123 – 154.*

Alshawi, H., Carter, D., Gamback, B. and Rayner, M. (1991). Translation by quasi logical form transfer. *Proceedings of the 29th Annual Meeting Assoc. Computational Linguistics, Berkeley, California.*

Alterman, R., Zito-Wolf, R. and Carpenter, T. (1998). Pragmatic Action. *Cognitive Science, Vol.22 (1), pp. 53 – 105.*

Ardissono, L., Barbero, C., Goy, A. and Petrone, G. (1999). An Agent Architecture for Personalized Web Stores. *Proceedings of the Third International Conference on Autonomous Agents, (Agents '99), Seattle.*

Artale, A., Franconi, E., Guarino, N. and Pazzi L. (1996). Part-Whole Relations in Object-Centered Formalisms: an Overview. *Data and Knowledge Engineering 20.*

Balabanovic, M. (1997). An adaptive web page recommendation service. *Proceedings of the first international conference on autonomous agents, pp. 378 – 385.*

Barwise, J. (1979). On branching quantifiers in English. *Journal of Philosophical Logic, 8, pp. 47 – 80.*

Bateman, J. A., Magnini, B. and Rinaldi, F. (1994). *The Generalized* Italian, German, English Upper Model. *Proceedings of the ECAI94 Workshop: Comparison of Implemented Ontologies, Amsterdam.*

Belew, R. K. (2001). A Cognitive Perspective on Search Engine Technology and the WWW. *The Cambridge University Press.*

Bhatia, S. K., and Deogun, J. S. (1998). Conceptual clustering in information retrieval. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, vol. 28, no. 3, pp. 427-436.*

Borgida, A. Extensible knowledge representation: The case of description reasoners. (1999). *Journal of Aritificial Intelligence, 10, pp. 399 – 434.*

Brachman, J. R., Borgida, A., Deborah L. M., Peter F. P., and Lori A. R. (1992). The CLASSIC Knowledge Representation System, or, KL-ONE: The Next Generation.

Proceedings of the 1992 International Conference on Fifth Generation Computer Systems, Tokyo, Japan.

Calvanese, D., Lenzerini, M. and Nardi, D. (1999). Unifying class-based representation formalisms. *Journal of Artificial Intelligence, 11, pp.199 – 240.*

Canamero, D. (1995). A knowledge-level approach to plan recognition. *Proceedings of the IJCAI'95 workshop on plan recognition. August, Montreal, Canada.*

Chandresekaran, B. and Josephson, R. John. (1999). What are ontologies, and why do we need them? *IEEE INTELLIGENT SYSTEMS, January/February, Vol.14, No.1.*

Charniak, E. (1986). A neat theory for marker passing. *AAAI-86.*

Charniak, E. (1995), Parsing with context free grammars and word statstics. *Department of Computer Science, Brown University, Technical Report.*

Charniak, E. (1995). Natural language learning. *ACM computing surveys, Vol 27.*

Charniak, E., Ge, N. and Hlae, J. (1998). A statastical approach to anaphora resolution. *Proceedings of the sixth workshop on very large corpora.*

Chaudri, V.K., Farquhar, A., Fikes, R., Karp, P.D. and Rice, J.P. (1998). Open Knowledge base Connectivity 2.0. *Knowledge Systems Laboratory, January 1998.*

Cocchiarella, N. B. (1991). Formal Ontology. *H. Burkhardt and B. Smith (eds.), Handbook of Metaphysics and Ontology. Philosophia Verlag, Munich, pp. 640-647.*

Cohn, A. G. (1987). Qualitative reasoning. *Lecture notes in AI, Vol.345, pp. 60 – 95.*

Corriveau, J. P. (1994). On the design of a concurrent object-oriented spreading activation architecture. *Proceedings of the 27th Annual Hawai International Conference on System Science. Vol.3, Information Systems: DSS/Knowledge-Based Systems (HICSS94 – 3). Wailea, HW, USA, pp. 73 – 81.*

Cottrell, G. W. and Small, S. L. (1983). A connectionist scheme for modelling word sense disambiguation. *Cognition and Brain Theory, 6(1), pp. 89 –120.*

Cowie, J. and Lehnert, W. (1996). Information Extraction. *Comm. ACM, Vol. 39, No.1, January, pp. 80--91.*

Craven, M., DiPasquo, D., Freitag, D., McCallum,A., Mitchell, T. Nigam, K. and Slattery, S. Learning to construct knowledge bases from the world wide web. *Artificial Intelligence, Vol.118, pp.69 – 113.*

Cree, G. S., McRae, K. and McNorgan, C. (1999). An attractor model of lexical conceptual processing: Simulating semantic priming. *Cognitive Science, Vol.23 (3), pp. 371 – 414.*

Crestani, F. (1997). Application of Spreading Activation Techniques in Information Retreival. *Artificial Intelligence Review, Vol.11(b), pp. 453 – 483.*

Croft, W. B. (1993). Knowledge-based and statistical approaches to text retrieval. *IEEE EXPERT, special series on AI in Text-Based Information Systems.*

Croft, W. B. and Lewis, D.D. (1987). An Approach to Natural Language Processing for Document Retrieval. *Proceedings of the Tenth Annual International ACMSIGIR Conference on Research and Development in Information Retrieval , ed. C.T. Yu and C.J. van Rijsbergen, New Orleans, pp. 26 – 32.*

Davis, R., Shrobe, H. and Szolovits, P. (1993). What is a knowledge representation. *AI Magazine, Vol.14 (1), pp. 17 – 33.*

Devedzic, V. (1999). Ontologies: Borrowing from software patterns. *Intelligence, Vol.10 (3), pp. 14 – 24.*

Farquhar, A., Frank, G. and Fikes, R. (1999). Building a large knowledge base from a structured source. *IEEE INTELLIGNT SYSTEMS, January/February, Vol.14, No.1.*

Farquhar, A., Fikes, R. and Rice, J. (1997). Tools For Assembling Modular Ontologies in Ontolingua. *Knowledge Systems Laboratory.*

Farquhar, A., and Fikes, R. (1997). Large-scale repositories of highly expressive reusable knowledge. *Technical Report, http://www.ksl.stanford.edu/KSL-Abstracts/KSL-97-02.html.*

Farquhar, A., Fikes, R. and Rice, J. (1996). The Ontolingua Server: A Tool for Collaborative Ontology Construction. *Knowledge Systems Laboratory, September, 1996.*

Farquhar, A., Fikes, R., Pratt, W. and Rice, J. (1995). Collaborative Ontology Construction for Information Integration. *Knowledge Systems Laboratory Department of Computer Science, KSL-95-63, August 1995.*

Fillmore, C. J. (1968). The case for case. *In Bach, E. W. and Harms, R. T. (Eds.), Universals in linguistic Theory, Holt, Rinehart & Winston, New York, pp. 1-88.*

Genesereth, M. R. and Treitel, R. (1987). Choosing Directions for Rules. *Journal of Automated Reasoning, pp. 395-431.*

Gomez-Perez, A. (1995). Some ideas and examples to evaluate ontologies. *Proceedings of the eleventh IEEE Conference on AI Applications. Editorial IEEE Computer Society Press, pp. 299 – 305.*

Gruber, T. R. (1993). A translation approach to portable ontologies. *Knowledge Acquisition, 5(2). pp. 199-220.*

Gruber, J. S. (1965). Studies in lexical relations. *Ph.D.thesis, MIT, Cambridge, MA.*

Guarino, N. and Welty, C. (2000a). Towards a methodology for ontology based model engineering. *Proceedings of the ECOOP-2000 Workshop on Model Engineering. June.*

Guarino, N. and Wetly, C. (2000b). Ontological analysis of taxonomic relationships. *Proceedings of the International Conference on Conceptual Modelling, October.*

Guarino, N. and Welty, C. (2000c). Identity, unity and individuality: Towards a formal toolkit for ontological analysis. *Proceedings of the European Conference on Artificial Intelligence, August, Amstredam.*

Guarino, N. and Wetly, C. (2000d). A formal ontology of properties. *Proceedings of the ECAI-2000 workshop on Applications of ontologies and problem-solving methods, August.*

Guarino, N. (1999a). OntoSeek: Content-based access to the web. *IEEE INTELLIGENT SYSTEMS, May/June, Vol.14, No.3.*

Guarino, N. (1999b). The role of identity conditions in ontology design. *Proceedings of the IJCAI-99 workshop on ontologies and problem-solving methods (KRR5), August, Stockholm, Sweden.*

Guarino, N. (1998). Formal ontology in information systems. *Proceedings of FOIS'98, Trento, Italy, June. Amsterdam, IOS Press, pp. 3 – 15.*

Guarino N. (1998). Some Ontological Principles for Designing Upper Level Lexical Resources. *Proceedings of the First International Conference on Lexical Resources and Evaluation, Granada, Spain, May, pp. 28-30.*

Guarino, N. (1997a). Understanding, Building, And Using Ontologies. *The International Journal of Human and Computer Studies, Vol.46, No.2/3, pp. 293 – 310.*

Guarino, N. (1997b). Some organizing principles for a unified top-level ontology. *The Working Notes of AAAI Symposium on Ontological Engineering, held in Stanford in March.*

Guarino, N. (1997c). Semantic Matching: Formal Ontological Distinctions for Information Organization, Extraction, and Integration. *Summer School on Information Extraction, Frascati, July.*

Guarino, N., Borgo, S. and Masolo, C. (1997). Logical Modelling of product knowledge: Towards a well-founded semantics for STEP. *Proceedings of the European Conference on Product Data Technology, Sophia Antipolis, France.*

Guarino, N. (1995). Formal ontology, conceptual analysis and knowledge representation. *The International Journal of Human and Computer Studies, special issue on the role of formal ontology in IT, Vol.43, N0.5/6.*

Guarino, N. and Giaretta, P. (1995). Ontologies and knowledge bases. *Towards a terminological clarification. IOS Press, Amsterdam.*

Guarino, N., Carrara, M. and Giaretta, P. (1994). Formalizing ontological commitments. *Proceedings of the AAAI.*

Guarino, N. (1993). The ontological level. *Proccedings of the 16th Wittgenstein Symposium, August, Kirchberg, Austria.*

Guarino, N. (1992). Concepts, attributes, and arbitrary relations. Some ontological criteria for structuring knowledge bases. *Data and Knowledge Engineering, Vol.8, pp. 249 – 261.*

Harabagiu, M. S. and Moldovan, D. (1999). A parallel system for textual inference. *In the IEEE Transactions Parallel and Distributed Systems, Vol.10, No.11.*

Harabagiu, S. M and Maldovan, I. D. (1996). A marker-propagation text understanding and inference system. *Proceedings of the FLAIRS – 96, May, Key West FL, pp. 55 –59.*

Hearst, M. and Hirsh H. (2000). AI's greatest trends and controversies. *IEEE INTELLIGENT SYSTEMS, Vol.15, No.1.*

Hirst, G. (1997). Context as a spurious concept. *The AAAI fall symposium on context in knowledge representation and natural language, Cambridge, Massachusetts.*

Hobbs, J. R. and Shieber, S. M. (1987). An algorithm for generating quantifier scopings. *Computational Linguistics, 13(1), pp. 47 - 55.*

Iwanska, L. M. and Saphiro, S. C. (2000). Natural language processing and knowledge representation. *AAAI.*

Jackendoff, R. (1972). Semantic interpretation in generative grammar. *MIT Press, Cambridge, MA.*

Jaszczolt, K. M. (1994). Default semantics, pragmatics and intentions. *Current Research in Semantic/Pragmatic Interfaces. Oxford: Elsevier Science, pp. 199 – 232.*

Johnson, M. L. and Fernandez-Duque, D. (1999). Attention metaphors: How metaphors guide the cognitive psychology of attention. *Cognitive Science, Vol.23 (1), pp. 83 –116.*

Johnson, M. L., (1987). The body in the mind: the bodily basis of meaning, imagination and reason. *Chicago: University of Chicago Press.*

Jurafsky, D. and Martin, J. H. (2000). Speech and language processing. *Prentice-Hall, inc.*

Knight, K. and Luk, S. (1994). Building a Large-Scale Knowledge Base for Machine Translation. *Proceedings of the National Conference on Artificial Intelligence (AAAI).*

Krovetz, R. and Croft, W. B. (1992). Lexical ambiguity and information retrieval. *ACM Transactions on Information Systems, 10(2), pp. 115 – 141.*

Lakoff, G. (1992). The contemporary theory of metaphor. *In Ortony, Andrew (ed). Metaphor and Thought (2nd edition), Cambridge University Press.*

Lakoff, G. (1987). Women, fire and dangerous things. *University of Chicago Press, Ltd., London.*

Lakoff, G. and Johnson, M. (1980). Metaphors We Live By. *University of Chicago Press.*

Leacock, C., Geoffrey, T. and Voorhes, E. (1993). Corpus-Based Statistical Sense Resolution. *Proceedings of the ARPA Human Language Technology Workshop.*

Lehman, J. F. (1994). Towards the essential nature of statistical knowledge in sense resolution. *Proceedings of the Twelfth National Conference on Artificial Intelligence, pp. 734-471.*

Lenat, D. B. (1998). The dimensions of context-space. www.cyc.com.

Lenat, D. B., Miller, G. and Yokoi, T. (1995). Cyc, Wordnet and EDR: Critiques and responses. *Communications of the ACM, Vol 38, No.11.*

Lenat, D. B. (1995). CYC: a large-scale investment in knowledge infrastructure. *Communications of the ACM, Vol.38, No.11.*

Lenat, D.B., Guha, R. V., Pittman, K., Pratt, D. and Shepperd, M. (1990). CYC: A midterm report. *Communications of the ACM, Vol.33, No.8.*

Luke, S., Spector, L., Rager, D. and Hendler, J. (1997). Ontology based web agents. *(Agents 1997). Assocation for Computing Machinery.*

Lyons, J. (1977). Semantics. *Newyork: Cambridge University Press.*

Mahesh, K. (1996). Ontology development for machine translation. *Technical Report MCCS 96-292, Computing Research Laboratory, New Mexico State University, Las Cruces, NM.*

Mahesh, K. Nirenberg, S. (1996). Meaning representation for knowledge sharing in practical machine translation. *Proceedings of the FLAIRS-96 track on Information Interchange, Florida AI Research Symposium, May.*

Mahesh, K. and Nirenburg, S. (1995a). A situated ontology for practical NLP. *Proceedings of the IJCAI-95 workshop on Basic Ontological Issues in Knowledge Sharing, August, Montreal, pp. 19 – 21.*

Mahesh, K. and Nirenburg, S. (1995b). Semantic classification for practical Natural Language Processing. *Proceedings Sixth ASIS SIG/CR Classification Research Workshop: An Interdisciplinary Meeting. October 8, Chicago IL.*

Martin, P. and Eklund, P. W. (2000). Knowledge retreival and the world wide web. *IEEE INTELLIGENT SYSTEMS, May/June, Vol.15, No.3.*

McAllester, D. and Givan, N. (1992). Natural language synatx and first-order inference. *Artificial Intelligence, 56.*

McCarthy, J. (1987). Generality in artificial intelligence. *Communications of the ACM, Vol.30, No.12.*

McCarthy, J. (1980). Circumscription – A form of non-monotonic reasoning. *Artificial Intelligence.*

McDermott, D., and Doyle, J., (1980). Non-monotonic logic. *Artificial Intelligence 13, pp. 41-72.*

Menzies, T. (1999). Cost benefits of ontologies. *Intelligence, Vol.10 (3), pp. 27 – 31.*

Miller, A. G., R., Fellbaum, C. (1998). Wordnet: An electronic lexical database. *MIT PRESS.*

Miller, G. A., Richard, B., Christiane, F., Derek, G. and Katherine J. M. (1990). Introduction to WordNet: an on-line lexical database. *International Journal of Lexicography 3 (4), pp. 235 - 244.*

Minsky, M. (2000). Commonsense-based interfaces. *Communications of the ACM, Vol.43, No.8.*

Minsky, M. (1989). K-Lines: A theory of memory. *Cognitive Science. Vol.4, pp. 117 – 133.*

Mooney, R. (1996). Comparative experiments on disambiguating word senses. An illustration of the role of bias in machine learning. *Proceedings of the Conference on Emperical Methods in Natural Language Processing. Philadelphia.*

Motta, E., Buckingham Shum, S. and Domingue, J (1999). Ontology-Driven Document Enrichment: Principles and Case Studies. *Proceedings of the KAW'99: 12th Banff Knowledge Acquisition Workshop, Banff, Canada, 1999, Dept. Computer Science, University of Calgary, CA .*

Mueller, T. E. (2000). A calendar with common sense. *Proceedings of the 2000 International Conference on Intelligent User Interfaces, New York: Association for Computing Machinery.*

Nebel, B. (2000). Knowledge Representation and Reasoning - The Theoretical Side of AI, in: *14th European Conference on Artificial Intelligence, Proceedings (ECAI 2000), Berlin, Germany, 2000, pp. 763.*

Newell, A. (1982). The Knowledge Level. *Artificial Intelligence. 18, pp. 87 - 127.*

Nirenburg, S., Raskin, V. and Onyshkevych, B. (1995). Apologiae Ontologiae. *Proceedings of the Conference on Theoretical and Methodological Issues in Machine Translation. Leuven, Belgium, July.*

Quillian, M. R. (1966). Semantic memory. *Unpublished doctoral dissertation, Carnegie Institute of Technology. Reprinted in part in M. Minsky [Ed.], Semantic Information Processing. Cambridge, Mass.: M.I.T. press, 1968.*

Reiter, R. (1980). A logic for default reasoning. Artificial Intelligence, *13, pp. 81-132.*

Resnik, P. (1999). Semantic similarity in a taxonomy: An information-based measure and its application to problems of ambiguity in natural language. *Journal of Artificial Intelligence, 11, pp. 95 – 130.*

Resinik, P. and Varian, H. R. (1997). Recommender systems. *Communiactions of the ACM, Vol.40, No.3.*

Ria, G. K. (1999). PHYSICIAN is a role played by an object, whereas SIGN is a role played by a concept. *Proceedings of the IJCAI-99 workshop on Ontologies and problem-solving methods (KRR5), Stockholm, Sweden.*

Rilof, E. (1993). Automatically constructing a dictionary for information extraction tasks. *Proceedings for the Eleventh National Conference on Artificial Intelligence 1993, AAAI Press, MIT Press, pp. 811 – 816.*

Russel, S. and Norvig, P. (1995). Artificial Intelligence, A modern approach. *Copyrighted by Prentice-Hall, Inc.*

Saba, W. and Raghuram, R. (2001). An experiment in language-based ontological design. *Submitted to AICS-2001, the 12th Irish Conference on Artificial Intelligence and cognitive Science.*

Saba, W. (2001). Towards an ontology of commonsense knowledge for natural language understanding. *Technical Report No. 01-006. School of Computer Science, University of Windsor. Also appears in Electronic Transactions on Artificial Intelligence (ETAI) as First Publication Archive.*

Saba, W. (2000). Plausible reasoning and the resolution of quantifier scope ambiguities. *Studia Logica – An International Journal of Symbolic Logic, 60.*

Saba, W. and Corriveau, J. P. (1999). Plausible reasoning in NLP without massive amount of background knowledge. *Third Australian Workshop on Commonsense reasoning.*

Saba, W. and Corriveau, J. P. (1997). A pragmatic treatment of quantification in natural language. *AAAI/IAAI-97, pp. 610 – 615.*

Sacco, G. M. (2000). Dynamic taxonomies: A model for large information bases. *IEEE transactions on knowledge and data engineering*, Vol.12, No.3.

Shapiro, S. C., McKay, D. P., Martins.J, and Morgado, E. (1981). SNePSLOG: A higher order logic programming language. *Presented at the Workshop on Logic Programming for Intelligent Systems, R.M.S. Queen Mary, Long Beach, CA.*

Shastri, L. (1993). A computational model of tractable reasoning – taking inspiration from cognition. *Proceedings of IJCAI-93, the Thirteenth International Joint Conference on AI, France, August/September 1993, pp. 202 – 207.*

Shastri, L. (1991). Why semantic networks? In Principles of Semantic Networks. *Edited by John Sowa, Morgan Kaufman Los altos.*

Shastri, L. (1989). Default reasoning in semantic networks: A formalization of recognition and inheritance. *Artificial Intelligence, 39, pp. 283 – 355.*

Shoham, Y. (1999). What we talk about when we talk about agents. *IEEE INTELLIGENT SYSTEMS, March/April 1999, Vol.14, No.2.*

Sowa, J. F. (1995). Knowledge representation, logical, philosophical and computational foundations. *The PWS Publishing Company, Boston.*

Stefik, M. J. and Smoliar, S. W. (1993). The Commonsense Reviews – Eight Reviews of: "Douglas Lenat and R.V. Guha, Building Large Knowledge-Based Systems: Representations and Inference in the CYC Project, Addison-Wesley 1990", and "Ernest Davis, Representations of Commonsense Knowledge, Morgan Kaufmann 1990", *Artificial Intelligence, Vol. 61, pp. 37 – 179.*

Touretzky, D. S. (1986). The mathematics of inheritance Systems. *Los Altos, California: Morgan kaufmann.*

Turing, A.M. (1950). Computing machinery and intelligence. *Mind, 59, pp. 433 – 460.*

Weinstien, P. and Alloway, G. (1997). Seed ontologies: growing digital libraries as distributed, intelligent systems. *Association for Computing Machinery, Digital Library 1997, pp. 83 – 90.*

Zadeh, L. A. and Yager, R. R. (1991). An Introduction to Fuzzy Logic Applications in Intelligent Systems. *Kluwer Academic Publishers..*

Zadeh, L. A (1989). Knowledge representation in fuzzy logic. *IEEE Transactions on Knowledge and Data Engineering, Vol.1, pp. 89-100.*

Zaiane, O. R., Fall, A., Rochefort, S., Dahl, V. and Tarau, P. (1997). Concept-based retreival using controlled natural language. *Natural Language Databases, Vancouver, British Columbia.*

Zadrozny, W. (1995). Context and ontology in understanding of dialogs. *Proceedings of the IJCAI'95 workshop on context in NL, August, Montreal.*

Zhu, X., Gauch, S., Gerhard, L. and Pretschner, A. (1999). Onotology-based web site mapping for information exploration. *ACM CIKM Conference on Information & Knowledge Management, Kansas City, Missouri.*

VITA AUCTORIS

The author was born in India in 1974. Apart from completing his BSc. degree in Software Engineering at the University of Windsor, Canada, in June 2000, he also completed the AHMA diploma in hotel management at the Hotel Institute of Montreux, Switzerland, in June 1994. He is currently a candidate for the MSc. degree in computer science at the University of Windsor, Windsor, Ontario, Canada, and hopes to graduate by September 2000.

His primary areas of research over the last few years have been natural language understanding, Ontologies, information retrieval and Mobile-agent computing. In the last six months however, he has also had the opportunity to explore parallel computing and virtual networks.