Word Sense Disambiguation

B. Tech. Project Final Report

Submitted in partial fulfillment of the requirements
for the degree of

Bachelor of Technology

by

Prithviraj B.P.
Roll No: 00005006

under the guidance of

Dr. Pushpak Bhattacharya

Department of Computer Science and Engineering
Indian Institute of Technology, Bombay
Mumbai

Contents

Acknowledgements

I would like to express my sincere gratitude towards my guide Dr. Pushpak Bhattacharya for his guidance and encouragement. I would also like to thank Ganesh Ramakrishnan for his valuable help.

Abstract

The task of word sense disambiguation (WSD) is to assign a sense label to a word in context. It has application in various NLP (Natural Language Processing) tasks such as machine translation, query-based information retrieval and information extraction. In this report we discuss a gloss based disambiguation system which uses WordNet glosses. The context of a word and the glosses of its different senses provided by WordNet prove helpful in disambiguating the word. The system is evaluated against Semcor and the results are presented in this report. We also present the notion of "soft WSD" in which the senses of a word are ordered according to their relevance in the context. The idea is that one should not commit to a particular sense of the word, but rather, to a *set of its senses*.

Chapter 1

Introduction

All languages use words with multiple meanings. The problem of determining the correct sense of a word, given the context, is called Word Sense Disambiguation (WSD). The meaning that a particular word takes, depends on the meanings that the surrounding words take, given that the surrounding words may also be ambiguous.

As a first example of ambiguity, consider the word "bank" and two of the senses that can be found in Webster's New Collegiate Dictionary :

1. the rising ground bordering a lake, river or sea.

2. an establishment for the custody, loan exchange, or issue of money, for the extension of credit, and for facilitating the transmission of funds.

This is perhaps the most famous example of an ambiguous word, but it is really quite atypical. More usually, a word has various somewhat related senses, and it is unclear whether to and where to draw lines between them. For example, consider *title*:

1. Name/heading of a book, statute, work of art or music, etc.

2. Material at the start of a film

3. The right of legal ownership (of land)

4. The document that is evidence of this right

5. An appellation of respect attached to a person's name

6. A written work

One can simply define the senses of a word to be its meanings as they are given in a particular dictionary, but this definition is often problematic since dictionaries differ in the number and kind of senses they list.

The problem of disambiguation, that is, determining which of the senses of an ambiguous word is invoked in a particular context, is of clear importance in many applications of natural language processing. A system for automatic translation from English to German, needs to translate "bank" as "Ufer" for first sense and as "Bank" for second sense. An information retrieval system answering a query about "financial banks" should only return documents that use "bank" in second sense. Whenever a system's action depends on the meaning of the text being processed, disambiguation is beneficial or even necessary.

The nature of ambiguity and disambiguation changes quite a bit depending on what training material is available. There are three main types of disambiguation based on different types of training material.

1. *Supervised disambiguation*: Here the disambiguation is based on a labelled training set. In training set the words are annotated with their contextually appropriate sense. This setting makes supervised disambiguation an instance of statistical classification: there is a training set of examples which are labelled as belonging to one of several classes (the senses). The task is to build a classifier which correctly classifies new cases.

2. *Dictionary-based disambiguation*: If we have no information about the sense categorization of specific instances of a word, we can fall back on a general categorization of the senses as given in dictionaries and in lexical resources. One simple algorithm can be to find the dictionary definitions of ambiguous words and find the sense which has maximum overlap (number of words shared between them) with the context words. This sense is the disambiguated sense. One can also use the thesaurus based disambiguation in which the basic inference is that the semantic categories of the words in a context determine the semantic category of the context as a whole, and that this category in turn determines which senses are to be used.

3. *Unsupervised disambiguation*: Generally available lexical resources or a small training set is all that the above methods require for disambiguation. Although this seems like a little to ask for, there are situations in which even such a small amount of information is not available. In particular, this is often the case in information retrieval. IR systems must be able to deal with text collections from any subject area, often very specialized ones. Lexical resources cover only general-purpose vocabulary and so are not useful for domain-specific collections. Completely unsupervised disambiguation is not possible if we mean sense tagging: an algorithm that labels occurrences as belonging to one sense or another. However, *sense discrimination* can be performed in a completely unsupervised fashion: one can discriminate between two different sets of objects without labelling them.

1.1 Organization

Chapter 2 looks at some of the existing algorithms for word sense disambiguation. The chapter includes information-theoretic disambiguation and Bayesian classification methods which come under the domain of supervised disambiguation, and Yarowsky's [10] dictionary based disambiguation method. Agirre's [4] disambiguation algorithm based on the concept of *conceptual density* is also included. Chapter 3 describes the widely used lexical resource, the English WordNet. English WordNet has been used extensively in many of the WSD algorithms. The chapter also contains a section about the recently released WordNet 2.0. We describe a new approach in WSD known as "Soft Word Sense Disambiguation" in chapter 4. In this methodology the senses of an ambiguous word are ordered according to their relevance in the sentence and one does not commit to a unique sense of the word. The soft WSD system is evaluated against semcor and results are also presented. We discuss a gloss based disambiguation algorithm in chapter 5. The results of evaluation against Semcor and the task given in Senseval-3 are included in the chapter. Finally we present the conclusions and future work in chapter 6.

Chapter 2

Various approaches for Word Sense Disambiguation

Word sense disambiguation is an important problem in computational linguistics. It has application in tasks like information retrieval and machine translation. This chapter gives a brief overview of different methods which have been applied for word sense disambiguation. We will study few methods like information-theoretic disambiguation and Bayesian classification, which come under supervised disambiguation. Yarowsky [10] proposes a solution to the problem of WSD using a thesaurus in a supervised learning environment. Word associations are recorded and for an unseen text, the senses of words are detected from the learnt associations. Agirre [4] uses a measure based on the proximity of the text words in WordNet (conceptual density) to disambiguate the words.

2.1 Supervised Disambiguation

A disambiguated corpus is available for training in the case of supervised disambiguation [3]. Each occurrence of the ambiguous word is annotated with its contextually appropriate sense. This setting makes supervised disambiguation an instance of statistical classification: there is a training set of examples which are labeled as belonging to one of several classes (the senses). The task is to build a classifier which correctly classifies new cases. We will describe two methods based on two very important concepts in statistical language processing: Information Theory and Bayesian Classification. The first approach looks at only one informative feature in the context, a structural feature. But this feature is carefully selected from a large number of potential "informants". The second approach treats the context of occurrence as a bag of words without structure, but it integrates information from many features.

2.1.1 Information-Theoretic Disambiguation

The information-theoretic algorithm tries to find a contextual feature that reliably indicates which of the senses of the ambiguous word is used. Some examples of indicators for French ambiguous words are listed in Table 2.1.1. For the verb *prendre* its object is a good indicator: *prendre une mesure* translates as to take a measure, *prendre une d'ecision* as to make a decision. Similarly, the tense of the verb *vouloir* and the word immediately to the left of *cent* are good indicators for these two words as shown in Table 2.1.1.

Table 2.1: Highly informative indicators for ambiguous French words.

ambiguous word	indicator	examples: value ->
prendre	object	*mesure* -> *to take* *decision* -> *to make*
vouloir	tense	present -> *to want* conditional -> *to like*
cent	word to the left	*per* -> *%* number -> *c*

Once the indicators for the words are found, disambiguation is simple.

- For the occurrence of the ambiguous word, determine the value of its indicator.
- Associate sense depending on the value of the indicator from the table.

The problem with this approach is that it is difficult to identify the indicators for each and every ambiguous word. Also it is difficult to identify indicators which are generic in nature.

2.1.2 Bayesian Classification

The essence of the Bayesian approach to disambiguation is to compute the probability of each of the senses of the ambiguous word given the context, $\Pr(s_i|C)$, and to choose the most probable sense. $\Pr(s_i|C)$ is computed using Bayes' Theorem:

$$\Pr(s_i|C) = \frac{\Pr(C|s_i)\Pr(s_i)}{\Pr(C)} \tag{2.1}$$

The denominator in the above equation can be ignored as we wish to maximize this quantity. We compute $\Pr(C|s_i)$ using the following independent assumptions:

$$\Pr(C|s_i) = \prod_{w \in C} \Pr(w|s_i) \tag{2.2}$$

$\Pr(w|s_i)$ and $\Pr(s_i)$ are computed as :

$$\Pr(w|s_i) = \frac{N(w, s_i)}{N(s_i)} \tag{2.3}$$

$$\Pr(s_i) = \frac{N(s_i)}{N(a)} \tag{2.4}$$

where $N(w, s_i)$ is the number of occurrences of w in a context of sense s_i in the training corpus, $N(s_i)$ is the number of occurrences of s_i in the training corpus, and $N(a)$ is the total number of occurrences of the ambiguous word a.

For each word in the sentence, the disambiguation algorithm finds out the score $(\Pr(C|s_i))$ for each sense of the ambiguous word a. The sense with the highest score is the disambiguated sense.

Training Data (Words in Context)
..uipment such as a hydraulic *shovel* capable of lifting 26 cubic..
..on .SB Resembling a power *shovel* mounted on a floating hul..
..uipment, valves for nuclear *generators*, oil-refinery turbines..
..1-penetrating carbide-tipped *drills* forced manufacturers to fi..
..nter of rotation .PP A rower *crane* is an assembly of fabricat..
..rshy areas .SB The crowned *crane*, however, occasionally..

Table 2.2: Concordance for words in the category TOOLS/MACHINERY

2.2 Thesaurus based Disambiguation

Thesaurus based disambiguation exploits the semantic categorization provided by a thesaurus like Roget's International Thesaurus. The basic inference is that the semantic categories of the words in a context determine the semantic category of the context as a whole, and that this category in turn determines which word senses are used.

Yarowsky [10] defines the sense of a word as the category listed for that word in Roget's International Thesaurus. Sense disambiguation will constitute selecting the listed category which is most probable given the surrounding context. Although this may appear to be crude approximation, it is surprisingly successful in partitioning the major senses of a word.

The main idea is based on the following three observations:

1. Different conceptual classes of words, such as ANIMALS or MACHINES tend to appear in recognizably different contexts.

2. Different word senses tend to belong to different conceptual classes. (*Crane*) can be an ANIMAL or a MACHINE.

3. If one can build a context discriminator for the conceptual classes, one has effectively built a context discriminator for the word senses that are members of those classes. Furthermore, the context indicators for a Roget category (eg. *gear, piston* and *engine* for the category TOOLS/MACHINERY will also tend to be context indicators for the members of that category (such as the machinery sense of *crane.*

Based on the above points the algorithm has three basic steps which are done for each of 1042 Roget Categories.

1. **Collect Contexts which are Representative of the Roget Category**

   The goal of this step is to collect a set of words that are typically found in the context of a Roget category. To do this, we extract concordance of 100 surrounding words for each occurrence of each member of the category in the corpus. Below is a sample set of partial concordance for words in the category TOOLS/MACHINERY.

   Although the concordance set should only include references to the given category, many spurious examples will also be there due to polysemy (such as *crane* in the above table). This level of noise can usually be tolerated but if one of these spurious senses was frequent

ANIMAL/INSECT
species(2.3), family(1.7), fish(2.4), animal (1.7),
wild(2.6), common(1.3), female(2.0),
inhabit(2.2), eat(2.2), nest(2.5),...
TOOLS/MACHINERY
tool(3.1), machine(2.7), engine(2.6), blade(3.8),
cut(2.6), saw(5.1), pump(3.5), device(2.2),
gear(3.5), wheel(2.8), shaft(3.3), wood(2.0),..

Table 2.3: weights of salient words for category TOOLS/MACHINERY and ANIMAL/INSECT

and dominated the set of examples, the situation could be disastrous. So if a word occurs k times in the corpus, all words in the context of that word contribute weight 1/k to frequency sums.

2. **Identify salient words in the collective context, and weight appropriately**

   A salient words is one which appears significantly more often in the context of a category than at other points in the corpus, and hence is a better than average indicator for the category. It can be formalized as : $\Pr(w|RCat)/\Pr(w)$, the probability of a word (w) appearing in the context of a Roget category divided by its overall probability in the corpus.

   Below are salient words for Roget categories ANIMAL/INSECT and TOOLS/MACHINERY. The numbers in parentheses are the log of the salience $\log(\Pr(w|RCat)/\Pr(w)))$, which we will refer to as the word's weight.

   The words in table 2.3 show the words which are likely to co-occur with the members of the category. The complete list for a category typically contains over 3000 words.

3. **Use the resulting weights to predict the appropriate category for a word in novel text**

   When any of the salient words derived in step 2 appear in the context of an ambiguous word, there is evidence that the word belongs to the indicated category. If several such words appear, the evidence is compounded. Using Bayes' rule, we sum their weights, over all words in context, and determine the category for which the sum is greatest. The context is defined to extend 50 words to the left and 50 words to the right of the polysemous word.

It is useful to look at one example in some more detail. Consider the following instance of *crane* and its contexts of 10 words to the right and the left.

.. lift water and to grind grain .PP Treadmills attached to **cranes** were used to lift heavy objects from Roman times, ...

Figure 2.1: An example sentence with ambiguous word *crane*.

TOOLS/MACHINE	Weight		ANIMAL/INSECT	Weight
lift	2.44		water	0.76
lift	2.44			
grain	1.68			
used	1.32			
heavy	1.28			
Treadmills	1.16			
attached	0.58			
grind	0.29			
water	0.11			
TOTAL	11.30		TOTAL	0.76

Table 2.4: Different categories for *crane*

The table 2.4 shows the strongest indicators identified for the two categories in the sentence above. The model weights, as noted above, are equivalent to $\log(\Pr(w|RCat)/\Pr(w))$. Several indicators were found for the TOOLS/MACHINERY class. There is very little evidence for the ANIMAL sense of *crane*, with the possible exception of *water*. The preponderance of evidence favors the former classification, which happens to be correct. The difference between the two scores indicate strong confidence in the answer:

The procedure described above is based on broad context models and hence works best only on words with senses which can be distinguished by their broad context. These are usually concrete nouns. The limitations are described below:

- **Topic Independent Distinctions**: One of the reasons that *interest* is disambiguated poorly is that it can appear in almost any context. While its "curiosity" sense is often indicated by the presence of an academic subject or hobby, the "advantage" sense (to be in one's interests) has few topic constraints. Distinguishing between two such abstractions is difficult. However, the financial sense of interest is readily identifiable and can be distinguished from the non-financial uses easily.

- **Minor Sense Distinctions within a Category**: Distinctions between the medicinal and narcotic senses of *drug* are not captured by the system because they both belong to the same Roget category (*remedy*).

- **Idioms**: The system is less successful in dealing with a word like *hand*, which is usually found in fixed expressions such as *on the other hand* and *close at hand*. These fixed expressions have more function than content, and therefore, they do not lend themselves to a method that depends on differences in content. This situation can be rectified easily as many idioms can be associated by simple lookup of a table storing idioms.

2.3 Disambiguation based on Conceptual Density

Agirre [4] presents a general automatic decision procedure for lexical ambiguity resolution based on a formula of the conceptual distance among concepts: Conceptual Density. Conceptual distance between two concepts is defined as the length of the shortest path that connects the concepts in a hierarchical semantic net. A measure of the relatedness among concepts can be a valuable prediction for several decisions in Natural Language Processing. For example, the relatedness of a certain word-sense to the context allows us to select that sense over the others, and actually disambiguate the word. Relatedness can be measured by a fine-grained conceptual distance among concepts in a hierarchical semantic net such as WordNet. This measure would allow to discover reliably the lexical cohesion of a given set of words in English.

The measure of conceptual distance among concepts should be sensitive to:

- the length of the shortest path that connects the concepts involved.
- the depth in the hierarchy: concepts in a deeper part of the hierarchy should be ranked closer.
- the density of concepts in the hierarchy: concepts in a dense part of the hierarchy are relatively closer than those in a more sparse region.
- the measure should be independent of the number of concepts we are measuring.

The conceptual density formula followed by Agirre compares areas of sub-hierarchies.

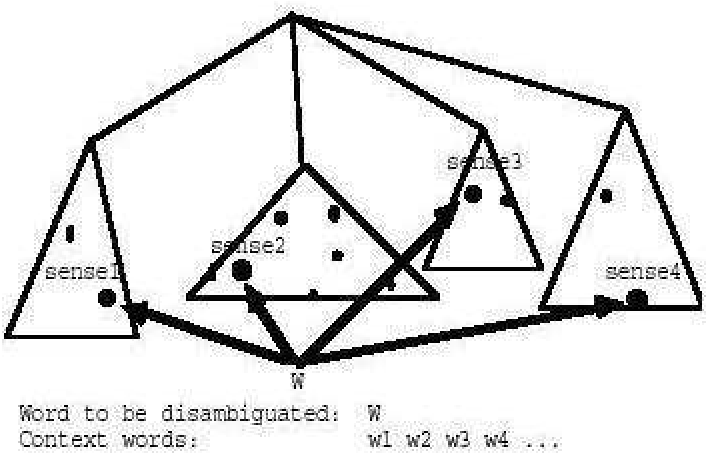

Figure 2.2: Senses of a word in WordNet

.

As an example of how Conceptual Density can help to disambiguate a word, in figure 2.2 the word W has four senses and several context words. Each sense of the words belongs to a sub-hierarchy of WordNet. The dots in the sub-hierarchies represent the senses of either the word to be disambiguated (W) or the words in the context. Conceptual Density will yield the highest density for the sub-hierarchy containing more senses of those, relative to the total amount of senses in the sub-hierarchy. The sense of W contained in the sub-hierarchy with highest Conceptual Density will be chosen as the sense disambiguating W in the given context. In figure 2.2, sense2 would be chosen.

Given a concept c, at the top of a sub-hierarchy, and given *nhyp* and h (mean number of hyponyms per node and height of the sub-hierarchy, respectively), the Conceptual Density for c when its sub-hierarchy contains a number m (marks) of senses of the words to disambiguate is given by the formula below:

$$CD(c,m) = \frac{\sum_{i=0}^{m-1} nhyp^i}{\sum_{i=0}^{h-1} nhyp^i} \tag{2.5}$$

The numerator expresses the expected area for a sub-hierarchy containing m marks (senses of the words to be disambiguated), while the divisor is the actual area, that is, the formula gives the ratio between weighted marks below c and the number of descendant senses of concept c. In this way, formula 2.5 captures the relation between the weighted marks in the sub-hierarchy and the total area of the sub-hierarchy below c.

nhyp is computed for each concept in WordNet in such a way as to satisfy equation 2.6, which expresses the relation among height, averaged number of hyponyms of each sense and total number of senses in a sub-hierarchy if it were homogeneous and regular:

$$descendants_c = \sum_{i=0}^{h-1} nhyp^1 \tag{2.6}$$

Thus, if we had a concept c with a sub-hierarchy of height 5 and 31 descendants, equation 2.6 will hold that *nhyp* is 2 for c. Conceptual Density weights the number of senses of the words to be disambiguated in order to make density equal to 1 when the number m of senses below c is equal to the height of the hierarchy h, to make density smaller than 1 if m is smaller than h and to make density bigger than 1 whenever m is bigger than h.

Given a window size, the program moves the window one word at a time from the beginning of the document towards its end, disambiguating in each step the word in the middle of the window and considering the other words in the window as context. The algorithm to disambiguate a given word w in the middle of a window of words W roughly proceeds as follows.

1. The algorithm first represents in a lattice the nouns present in the window, their senses and hypernyms.

2. Then the program computes the Conceptual Density of each concept in WordNet according to the senses it contains in its sub-hierarchy.

3. It selects the concept c with highest density.

4. Selects the senses below it as the correct senses for the respective words.

If a word from W:

- has a single sense under c, it has already been disambiguated.

- has not such a sense, it is still ambiguous.

- has more than one such senses, we can eliminate all the other senses of w, but have not yet completely disambiguated w.

The algorithm proceeds then to compute the density for the remaining senses in the lattice, and continues to disambiguate words in W (back to steps 2, 3 and 4). When no further disambiguation is possible, the senses left for w are processed and the result is presented.

We will illustrate the algorithm through an example. Consider the following text: *The jury(2) praised the administration(3)and operation(8) of the Atlanta Police_Department(1), the-Fulton_Tax_Commissioner_'s_Office, the Bellwood and Alpharetta prison_farms(1),Grady_Hospital and the Fulton_Health_Department.*

The underlined words are nouns represented in WordNet with the number of senses between brackets. The noun to be disambiguated in our example is *operation*, and a window size of five will be used.

- **(Step 1):** Figure 2.3 shows partially the lattice for the example sentence.

```
police_department_0
   => local department, department of
      local government
      => government department
         => department
         jury_1, panel
            => committee, commission
         operation_3, function
            => division
               => administrative unit
                  => unit
                     => organization
                        => social group
                           => people
                              => group

administration_1, governance...
jury_2
   => body
      => people
         => group, grouping
```

Figure 2.3: Partial lattice for the example sentence

.

  Word senses to be disambiguated are shown in bold. Underlined concepts are those selected with highest Conceptual Density. Monosemic nouns have sense number 0.

- **(Step 2):**`<administrative_unit>` for instance, has underneath 3 senses to be disambiguated and a sub-hierarchy size of 96 and therefore gets a Conceptual Density of 0.256. `<body>` with 2 senses and sub-hierarchy size of 86 gets 0.062.

- **(Step 3):**`<administrative_unit>`, being the concept with highest Conceptual Density is selected.

- **(Step 4):Operation_3, police_department_0** and **jury_1** are the sense chosen for *operation, Police_department* and *jury*. All the other concepts below <administrative_unit> are marked so that they are no longer selected. Other senses for the above words are deleted e.g **jury_2**. In the next loop of the algorithm <body will have only one disambiguation-word below it, and therefore its density will be much lower. At this point the algorithm detects that further disambiguation is not possible and quits the loop.

The important feature of Agirre's work is that the method presented above is ready-usable in any general domain and on free-running text, given parts of speech tags. It does not need any training and uses word sense tags from WordNet.

Chapter 3

Introduction to English WordNet

English WordNet [6], also known as Princeton WordNet, is a lexical database motivated by the principles of human mental lexicon. It was developed in 1985 at Princeton University under the guidance of Prof. George Miller. Its design was inspired by the psycholinguistic theories of human lexical memory. The most important feature of the English WordNet is that it organizes lexical information in terms of word meaning, rather than word form. So it can also be used as a thesaurus.

3.1 Lexical Matrix

A word is an association between a word form that plays a syntactic role and a concept. This leads to the concept of lexical matrix. Word forms are listed as headings for columns while word meanings are the headings for rows. Entry in a cell implies that the form in that column can be used to represent the meaning in that row. In figure 3.1 the entry $E_{1,1}$ implies that the word form F_1 in that column can be used to represent the word meaning M_1. If there are two entries in the same column it means that the same word form represents more than one meaning, so it is a polysemous word. If there are two entries in a row, the two word forms have the same meaning and hence are synonymous.

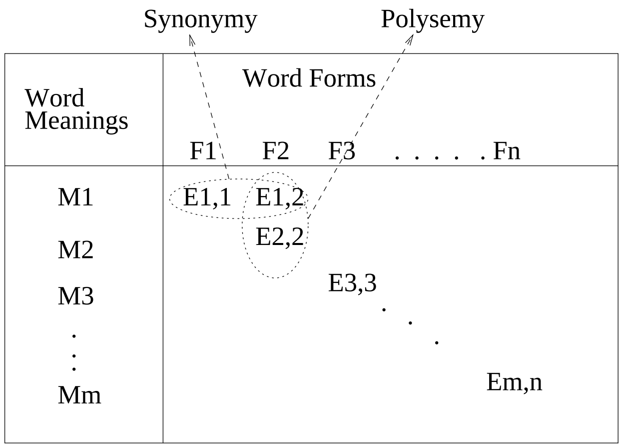

Figure 3.1: Lexical Matrix

The lexical matrix is represented in WordNet by mapping between written words and synsets.

A set containing synonyms of a word is known as a synset. It represents the meaning of the word. For example, {*board, plank*} and {*board, committee*} are synsets of board. Each synset denotes a different meaning. The members of synset are related by the relation synonymy.

3.2 Relations in English WordNet

There are two kinds of relations in English WordNet:

1. Semantic Relations: They are based on the meaning/concept of the word, rather than on its syntactic form.

2. Lexical Relations: They are based on the syntactic form of the word.

English WordNet is organized by relations between synsets. The following relations are used to organize different parts of speech.

1. **Synonymy** : It is the most important relation in WordNet, since it is used to represent the lexical matrix. Two words are synonymous in a linguistic context C if the substitution of one for the other in C does not change the truth value. For example, the substitution of *plank* for *board* will not alter the truth values in carpentry contexts, although there are other contexts of *board* where that substitution would be totally inappropriate like *committee*.

2. **Antonymy** : Given two words which are antonyms of each other, in a word association test the most common response to one of the word is the other word. For example, most people will respond with "victory" when they hear "defeat" and vice versa. This implies that antonymy is an important relation used by human beings for linguistic knowledge. It is represented in WordNet as below. The synsets *man* and *woman* would be like
   { *[man,woman !], person,*`@`*...*(a male person)}
   { *[man,woman !], person,*`@`*...*(a female person)}

   The symmetric relation of antonymy is represented by the ! pointer. The square bracket indicates that antonymy is a lexical relation between words, rather than a semantic relation between concepts. For example, the meanings {*rise, ascend*} and {*fall, descend*} may be conceptual opposites, but they are not antonyms; *rise/descend* or *ascend/fall* look odd as antonyms.

3. **Hypernymy-Hyponymy** : Hypernymy is a semantic relation between word meanings to capture superset-hood. Similarly, hyponymy is the semantic relation to capture subset-hood. For example, *maple* is a hyponym of *tree*, and *tree* is a hypernym of *maple*. A concept represented by the synset {`x, x',...`} is said to be a hyponym of the concept represented by the synset {`y, y',...`} if sentences can be constructed from such frames as *An x is a (kind of) y*.

4. **Meronymy**: The part-whole (or HAS-A) relation is known as meronymy. A concept represented by the synset { x_1, x_2, ...} is a meronym of a concept represented by the synset { y_1, y_2, ...} if native speakers of the language accept sentences constructed from

such frames as *A y has an x (as a part)* or *An x is a part of y*. For example *beak* and *wing* are meronyms of *bird*. The relation has an inverse: if W_m is a meronym of W_h then W_h is said to be a holonym of W_m. *Bird* is said to be a holonym of *beak*.

The general definition of Meronymy that is given above is not a reliable test of Meronymy. In many instances the transitivity property fails if we go by this definition. For example, *handle* is a meronym of *door* and *door* is a meronym of *house*, but it sounds odd to say *The handle is a part of the house*. Consider another example, *The branch is a part of the tree* and *The tree is a part of the forest* do not imply that *The branch is a part of the forest* because the branch/tree relation is different from the tree/forest relation. Such observations raise questions about how many different kinds of "part of" relations are there. Three types of these meronymy relations are included in English WordNet

(a) W_m #p-> W_h indicates that W_m is a component part of W_h .

(b) W_m #m-> W_h indicates that W_m is a member of W_h .

(c) W_m #s-> W_h indicates that W_m is the stuff that W_h is made from.

Of these three, the "is a component" relation '#p' is the most frequent.

3.3 WordNet 2.0

WordNet 2.0 has been released recently by Princeton university. It contains many new relations which were not present in WordNet 1.7.1. The following new links have been added:

- **Derivational Morphology**: It is a cross part-of-speech link between nouns and verbs. It links the synsets of verbs which have been derived from noun-forms to their corresponding noun synsets and vice versa.

```
Derived Forms of noun shout
1 sense of shout
Sense 1
cry, outcry, call, yell, shout, vociferation
       RELATED TO->(verb) shout#3
           => exclaim, cry, cry out, outcry, call out, shout
       RELATED TO->(verb) shout#2
           => shout, shout out, cry, call, yell, scream, holler, hollo, squall
       RELATED TO->(verb) shout#1
          => shout
```

Figure 3.2: Derivational link for noun *shout*

In figure 3.2 we see that the noun *shout* is linked to three synsets of verb *shout* implying that it is derived from these synsets. These relations are two-way i.e. the three senses of verb *shout* are linked to this single sense of noun *shout*. There are 42,000 such derivational links in WordNet 2.0.

- **Domain Classification**: Some synsets have been classified into domains. There is a link from a topical synset to a term assigned to its domain.

```
Derived Forms of verb shout
3 of 4 senses of shout
Sense 1
shout
       RELATED TO->(noun) shout#1
           => cry, outcry, call, yell, shout, vociferation
       RELATED TO->(noun) shouter#1
           => roarer, bawler, bellower, screamer, screecher, shouter, yeller

Sense 2
shout, shout out, cry, call, yell, scream, holler, hollo, squall
       RELATED TO->(noun) shout#1
           => cry, outcry, call, yell, shout, vociferation
       RELATED TO->(noun) shouting#2
           => yelling, shouting

Sense 3
exclaim, cry, cry out, outcry, call out, shout
       RELATED TO->(noun) shout#1
           => cry, outcry, call, yell, shout, vociferation
```

Figure 3.3: Derivational link for verb *shout*

```
Domain of verb diverge
1 of 4 senses of diverge
Sense 2
diverge
       CATEGORY->(noun) mathematics#1, math#1, maths#1
```

Figure 3.4: Domain link for verb *diverge*

The sense 2 of verb *diverge* which is "have no limits as a mathematical series" comes under the domain of mathematics as shown in figure 3.4.

Chapter 4

Soft Word Sense Disambiguation

Word sense disambiguation is defined as the task of finding *the* sense of a word in a context. The idea that one should not commit to a particular sense of the word, but rather, to a *set of its senses* which are not necessarily orthogonal or mutually exclusive, is explored in this chapter. Very often, WordNet gives for a word multiple senses which are related and which *help connect* other words in the text. This observation is referred to as the relevance of the sense in that context. Therefore, instead of picking a single sense, the senses are ranked according to their relevance to the text. As an example, consider the usage of the word *bank* in figure 4.1. In WordNet, *bank* has 10 noun senses. The senses which are relevant to the text are shown in figure 4.2.

A passage about some *bank* A Western Colorado bank with over $320 Million in assets, was formed in 1990 by combining the deposits of two of the largest and oldest financial institutions in Mesa County

Figure 4.1: One possible usage of *bank* as a *financial_institution*

Relevant senses

1. *depository financial institution, bank, banking concern, banking company :* a financial institution that accepts deposits and channels the money into lending activities; "he cashed a check at the bank"; "that bank holds the mortgage on my home"

2. *bank, bank building :* a building in which commercial banking is transacted; "the bank is on the corner of Nassau and Witherspoon"

3. *bank :*(a supply or stock held in reserve for future use (especially in emergencies))

4. *savings bank, coin bank, money box, bank:* (a container (usually with a slot in the top) for keeping money at home; "the coin bank was empty")

Figure 4.2: Some relevant senses for *bank*

These senses are ordered according to their relevance in this context. It is apparent that the first two senses have equal relevance. The applicability of the senses tapers off as we move down the list. This example motivates soft sense disambiguation [8]. *Soft sense disambiguation* is defined as the process of enumerating the senses of a word in a ranked order. This could be an end in itself or an interim process in an IR task like question answering.

4.1 Approach to soft WSD

A completely probabilistic framework for word-sense disambiguation with a semi-supervised learning technique utilizing WordNet, is described by [9]. The relevance of senses of a word are probabilistically determined through a Bayesian Belief Network.

In general, there could be multiple words in the document that are caused to occur together by multiple hidden concepts. This scenario is depicted in figure 4.3. The causes themselves may have hidden causes.

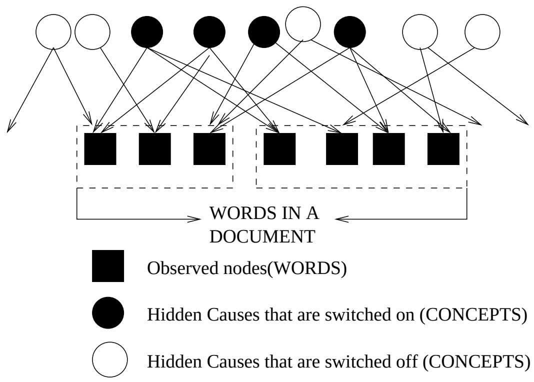

Figure 4.3: Motivation

These causal relationships are represented in WordNet which encodes relations between words and concepts (synsets). For instance WordNet gives the *hypernymy* relation between the concepts { animal} and { bear}.

4.1.1 Inferencing on lexical relations

It is difficult to link words to appropriate synsets in a lexical network in a principled manner. On the example of *animal* and *bear*, the English WordNet has five synsets on the path from *bear* to *animal*: {carnivore...}, {placental_mammal...}, {mammal...}, {vertebrate..}, {chordate...}. Some of these intervening synsets would be extremely unlikely to be associated with a corpus that is not about zoology; a common person would more naturally think of a *bear* as a kind of animal, skipping through the intervening nodes.

Clearly, any scoring algorithm that seeks to utilize WordNet link information must also *discriminate* between them based (at least) on usage statistics of the connected synsets. Also required is an estimate of the likelihood of instantiating a synset into a token because it was *activated* by a closely related synset. We find a Bayesian belief network (BBN) a natural structure to encode such combined knowledge from WordNet and corpus (for training).

4.1.2 Building a BBN from WordNet

Our model of the BBN is that each synset from WordNet is a boolean *event* associated with a word. Textual tokens are also events. Each event is a node in the BBN. Events can *cause* other events to happen in a probabilistic manner, which is encoded in Conditional Probability Tables. The specific form of CPT we use is the well-known *noisy-OR* for the words and *noisy-AND* for the synsets. This is because a word is *exclusively* instantiated by a cluster of parent synsets in the BBN, whereas a synset is compositionally instantiated by its parent synsets. The noisy-OR and noisy-AND models are described in [7].

We introduce a node in the BBN for each noun, verb, and adjective synset in WordNet. We also introduce a node for each token in the corpus. Hyponymy, meronymy, and attribute links are introduced from WordNet. *Sense links* are used to attach tokens to potentially matching synsets. For example, the string "flag" may be attached to synset nodes {sag, droop, swag, flag} and {a conspicuously marked or shaped tail}. (The purpose of probabilistic disambiguation is to estimate the probability that the string "flag" was *caused* by each connected synset node.)

This process creates a hierarchy in which the parent-child relationship is defined by the semantic relations in WordNet. A is a parent of B iff A is the *hypernym* or *holonym* or *attribute-of* or A is a synset containing the word B. The process by which the BBN is built from WordNet graph of synsets and from the mapping between words and synsets is depicted in figure 4.4. We define *going-up* the hierarchy as the traversal from child to parent.

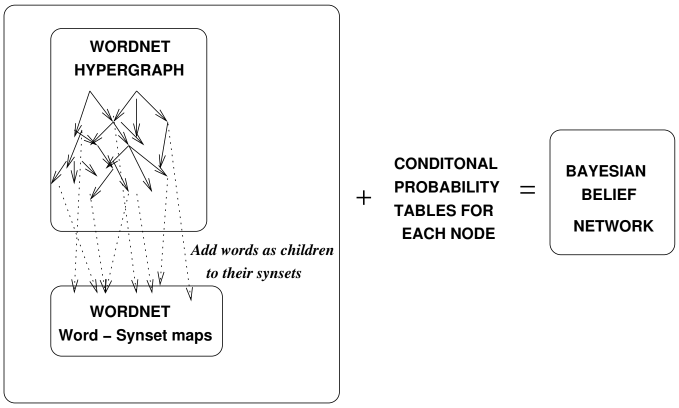

Figure 4.4: Building a BBN from WordNet and associated text tokens.

4.1.3 Training the belief network

The figure 4.5 describes the algorithm for training the BBN obtained from the WordNet. We initialize the CPTs as described in the previous section. The instances we use for training are windows of length M each from the untagged corpus. Since the corpus is not tagged with WordNet senses, all variables, other than the words observed in the window (i.e. the synset nodes in the BBN) are hidden or unobserved. Hence we use the Expectation Maximization algorithm [5] for parameter learning. For each instance, we find the expected values of the hidden variables, given the "present" state of each of the observed variables. These expected values are used after each pass through the corpus to update the CPT of each node. The iterations through the corpus are done till the successive iterations do not differ more than a small threshold. In this way we customize the BBN CPTs to a particular corpus by learning the local CPTs.

4.2 The WSD algorithm: ranking word senses

Given a passage, we clamp the BBN nodes corresponding to words, to a state of 'present' and infer using the network, the score of each of its senses which is the probability of the corresponding synset node being in a state of "present". For each word, we rank its senses in

while CPTs do not converge **do**
  for each window of M words in the text **do**
    Clamp the word nodes in the Bayesian Network to a state of 'present'
    for each node in Bayesian network **do**
      find its joint probabilities with all configurations of its parent nodes (E Step)
    end for
  end for
  Update the conditional probability tables for all random variables (M Step)
end while

Figure 4.5: Training the Bayesian Network for a corpus

Load the Bayesian Network parameters
for each passage p **do**
  clamp the variables (nodes) corresponding to the passage words $(w_1, w_2 ... w_n)$ in network to a state of 'present'
  Find the probability of each sense of each word, being in state 'present' i.e., $\Pr(s|w_1, w_2 .. w_n)$
end for
Report the word senses of each word, in decreasing order of ranks.

Figure 4.6: Ranking word senses

decreasing order of its score. In other words, the synset given the highest rank (probability) by this algorithm becomes the most probable sense of the Word.

4.3 Evaluation

We evaluated the above algorithm by running it on Semcor 1.7.1 corpus [2]. Semcor corpus is a subset of the famous Brown corpus [1] sense-tagged with WordNet 1.7.1 synsets. The soft WSD system produces rank ordered synsets on the semcor words (at most two senses). We show below in figure 4.7 the output of the system for the word *study*. Both semcor's tag and soft WSD system's first tag are correct, though they differ. The second tag from soft WSD system has low weightage and is wrong in this context. The synsets marked with ** represent the correct meaning.

Next we present an example of the second marking of the sense being correct. The word in question is the verb *urge* (figure 4.8).

Table4.1 summarizes soft WSD results obtained by us. If the first meaning given by the soft WSD system is correct then it is counted towards the *first match*; similarly for the *second match*.

Although the accuracy of the system is not very good, the main contribution is that it employs a completely probabilistic approach for word-sense disambiguation with a semi-supervised learning technique utilizing WordNet.

Passage from Semcor It recommended that Fulton legislators act to have these laws *studied* and revised to the end of modernizing and improving them.

Semcor tag: [Synset: [Offset: 513626] [POS: verb] Words: analyze, analyse, study, examine, canvass –(consider in detail and subject to an analysis in order to discover essential features or meaning; "analyze a sonnet by Shakespeare"; "analyze the evidence in a criminal trial"; "analyze your real motives")]

soft WSD tags: **[Synset: study 0 consider 0 [Gloss =] : give careful consideration to; "consider the possibility of moving" [Score = 0.62514]]

[Synset: study 4 meditate 2 contemplate 0 [Gloss =] : think intently and at length, as for spiritual purposes; "He is meditating in his study" [Score = 0.621583]]

Figure 4.7: Example of *first match* with Semcor's marking

Passage from Semcor It *urged* that the city take steps to remedy this problem.

Semcor tag: Synset: [Offset: 609547] [POS: verb] Words: urge, urge_on, press, exhort –(force or impel in an indicated direction; "I urged him to finish his studies")

soft WSD tags: [Synset: cheer 1 inspire 1 urge 1 barrack 1 urge_on 1 exhort 1 pep_up 0 [Gloss =] : urge on or encourage esp. by shouts; "The crowd cheered the demonstrating strikers" [Score = 0.652361]]

**[Synset: recommend 1 urge 3 advocate 0 [Gloss =] : push for something;"The travel agent recommended strongly that we not travel on Thanksgiving Day" [Score = 0.651725]]

Figure 4.8: Example of *second match* being correct

Table 4.1: Results of soft WSD

Total ambiguous nouns	139
Nouns first match	66
Nouns second match	46
Total ambiguous verbs	67
verbs first match	24
verbs second match	23

Chapter 5

Gloss based Algorithm for Disambiguation

We have formulated a gloss (WordNet glosses) based algorithm for disambiguation of words. Different types of glosses, based on different types of relations in WordNet like *hypernymy, holonymy etc.* are used. The main idea behind this approach is to use the context to find the correct sense of the word using its gloss.

5.1 Gloss

Querying WordNet for the noun *boy* gives the following output:

Senses of *boy* The noun boy has 4 senses (first 4 from tagged texts)

1. male child, boy – (a youthful male person; "the baby was a boy"; "she made the boy brush his teeth every night"; "most soldiers are only boys in uniform").
2. boy – (a friendly informal reference to a grown man; "he likes to play golf with the boys").
3. son, boy – (a male human offspring; "their son became a famous judge"; "his boy is taller than he is").
4. boy – (offensive term for Black man; "get out of my way boy").

Figure 5.1: Noun senses for *boy*

The entry for *boy* sense no.1 has synonyms - { *male child, boy* }, gloss - { *a youthful male person* } and examples - { *the baby was a boy, she made the boy brush his teeth every night, most soldiers are only boys in uniform* }. The gloss in our algorithm would refer to all these 3 entries. So for example our gloss for *male#n#1 (n is the part of speech and 1 denotes the sense no.)* would be the set of words - { *male child boy a youthful male person the baby was a boy she made the boy brush his teeth every night most soldiers are only boys in uniform* }.

5.2 Types of Gloss

There can be different types of glosses depending on the relations in WordNet.

1. *Lesk :* These glosses contain the synonyms, examples and the WordNet gloss of a sense of the word and the same attributes of its immediate hypernym. Consider the sense 3 of *boy*,

   son, boy – (a male human offspring; "their son became a famous judge"; "his boy is taller than he is")
   => male offspring, man-child – (a child who is male)

   Lesk gloss for sense 3 of noun *boy* would be - {*son boy male human offspring their became a famous judge his is taller than he man-child child who* }

2. *Lin :* They contain the synonyms of the word together with its hypernyms. Consider the sense 3 of noun *boy*,

```
Sense 3
son, boy
    => male offspring, man-child
       => child, kid
          => offspring, progeny, issue
             => relative, relation
                => person, individual, someone, somebody, mortal, human, soul
                   => organism, being
                      => living thing, animate thing
                         => object, physical object
                            => entity, physical thing
                   => causal agent, cause, causal agency
                      => entity, physical thing
       => male, male person
          => person, individual, someone, somebody, mortal, human, soul
             => organism, being
                => living thing, animate thing
                   => object, physical object
                      => entity, physical thing
             => causal agent, cause, causal agency
                => entity, physical thing
```

Figure 5.2: Hypernyms of boy#n#3

.

   So the *Lin* gloss for boy#n#3 is - {*son boy male offspring male-child child kid offspring progeny issue relative relation person individual someone somebody mortal human soul organism being living thing animate object physical entity causal agent cause agency person*}

3. *Lin-Lesk-hyper :* It contains both the *Lin* and the *Lesk* gloss for a word.

4. *Lin-Lesk-Holo :* It contains the *Lin* gloss, *Lesk* gloss and the Holonyms for a word.

5.3 Main Algorithm

The basic idea is to find the content words in the context of the ambiguous word and then find their intersection(common words in the context and the gloss) with the gloss of each sense of the word. The scores are based on the intersections. The senses are then ordered with respect to their scores. So soft word sense disambiguation is done.

During initialization we first find the frequency of the words occurring in the WordNet glosses. The inverse document frequency (idf) is taken as the inverse of the frequency of the words. Now given a document we take a window of 1 sentence or more from it. In this window we select one word at a time and treat the rest of the words as context words. The context can be taken as it is or they also can be expanded to their glosses. Intersection is found between this set of words and the gloss of each sense of the word to be disambiguated. The score is found from the idf of the common words. The senses are given out in their order of scores. There are several parameters which can be changed in the algorithms. We discuss them below.

5.3.1 Parameters

The algorithm has several parameters and each one has an influence on the result.

1. **GlossType :** This shows the type of gloss being used in the algorithm. It can be *lin, lesk, lin-lesk-hyper* or *lin-lesk-holo*.

2. **Stemming :** Sometimes the words in the context are related semantically with the gloss of the ambiguous word but they may not be in the same morphological form. For example, suppose that the context contains the word *Christian* but the gloss of the word contains the word *Christ*. The base form of both the words is *Christ* but since they are not in the same morphological form they will not be treated as common words during intersection. Stemming of words may prove useful in this case as after stemming both will give the same base form.

3. **FullContextGlossExpansion :** It shows whether the gloss of the context words should be taken or not. If set true then the gloss of all the senses of the context words would also be included for intersection.

4. **WindowSize :** The window size can be 1 sentence, 2 sentence etc and may also be 1 paragraph, 2 paragraph etc. It shows the total context window. The words to be disambiguated are taken from this window one by one while the rest of the words serve as context word for the ambiguous word.

5.4 Experimental Results

The program was evaluated against Semcor and was also used in Senseval-3 competition. We present the results in this section.

5.4.1 Results for Semcor

For experiments we chose the Semcor 1.7 corpus. It has been manually tagged using WordNet 1.7 glosses. ReRank1 denotes the percentage of cases where the highest scoring sense is the correct sense while ReRank2 denotes the percentage of cases where out of the first two highest senses one is correct. Note that we take the first sense of the word if the score is 0 for all the senses.

Stemming	WindowSize	FullGloss	POS	ReRank1	ReRank2	Total Words
No	1 Sent	true	n	50.3%	69.2%	81891
No	1 Sent	true	v	29.1%	50.1%	83545
No	1 Sent	false	n	71.4%	83.9%	34952
No	1 Sent	false	v	41.5%	64.7%	17068
No	2 Sent	true	n	47.7%	66.8%	38662
No	2 Sent	true	v	26.4%	44.8%	20641
No	2 Sent	false	n	49.1%	67.7%	3397
No	2 Sent	false	v	24.9%	41.4%	1593
No	3 Sent	false	n	47.3%	66.5%	6954
No	3 Sent	false	v	25.5%	41.6%	3421

Table 5.1: Results for *Lin* glosses

5.4.2 Results for Senseval-3 task

Senseval is an online competition for evaluation of the strengths and weaknesses of WSD programs with respect to different words and different languages. The third Senseval competition (Senseval-3) is taking place currently. The task we attempted was disambiguation of WordNet glosses. The input was given in xml format and was pos-tagged. We used *Lesk* glosses and sentence window size of 1 sentence. The results are presented in the table 5.5.

The results of our gloss based disambiguation system show that an optimal configuration of the parameters is essential to get good results. Most of the time *lesk* glosses together with stemming give better results than other. But it may be worthwhile to find out the weightage for different types of glosses and use all of them together. The reason behind some of the high scores is that when there are no common words between the gloss and the context words, the score is zero and so the first sense(which is the most frequently used sense) is taken as the correct sense.

Stemming	WindowSize	FullGloss	POS	ReRank1	ReRank2	Total Words
Yes	1 Sent	true	n	62.2%	80.32%	49245
Yes	1 Sent	true	v	36.6%	59.5%	83545
No	2 Sent	true	n	57.04%	77.21%	77746
No	2 Sent	true	v	34.2%	56%	80994
Yes	2 Sent	true	n	45.8%	65.8%	77746
Yes	2 Sent	true	v	22.8%	40%	47520
Yes	2 Sent	false	n	58.13%	78.04%	51033
Yes	2 Sent	false	v	34.03%	56%	27558
Yes	3 Sent	false	n	54.7%	76.3%	6132
Yes	3 Sent	false	v	31.4%	51%	3026
Yes	3 Sent	true	n	47.7%	66.1%	1755
Yes	3 Sent	true	v	24.4%	42.5%	827

Table 5.2: Results for *Lesk* glosses

Stemming	WindowSize	FullGloss	POS	ReRank1	ReRank2	Total Words
No	1 Sent	true	n	43%	61.5%	36014
No	1 Sent	true	v	21.4%	35.8%	17705
Yes	1 Sent	true	n	41.3%	59.3%	7676
Yes	1 Sent	true	v	21.1%	36%	3651
No	2 Sent	false	n	53.6%	74.9%	4203
No	2 Sent	false	v	29.7%	50.6%	2032
No	3 Sent	false	n	50.9%	73.1%	3694
No	3 Sent	false	v	29%	47.8%	1796

Table 5.3: Results for *lin-lesk-hyper* glosses

Stemming	WindowSize	FullGloss	POS	ReRank1	ReRank2	Total Words
No	1 Sent	true	n	49.18%	71.5%	8004
No	1 Sent	true	v	26.37%	43.8%	3860
No	2 Sent	false	n	62.75%	79.7 %	23938
No	2 Sent	false	v	37.5%	58.6 %	10862
No	2 Sent	true	n	48.2%	73.2%	4051
No	2 Sent	true	v	26%	43.3%	1947
No	3 Sent	true	n	48.5%	74.3%	2886
No	3 Sent	true	v	25%	43.5%	1372
No	3 Sent	false	n	61.08%	77.75%	5737
No	3 Sent	false	v	35.6%	54.7%	2815

Table 5.4: Results for *lin-lesk-holo* glosses

Stemming	WindowSize	FullGloss	POS	ReRank1	ReRank2	Total Words
Yes	1 Sent	false	n	72.9%	88.5%	20244
Yes	1 Sent	false	v	43.5%	62%	5235
Yes	1 Sent	true	n	65.1%	83%	19547
Yes	1 Sent	true	v	26.2%	44.07%	5051

Table 5.5: Senseval-3 task of disambiguation of WordNet glosses

Chapter 6

Conclusion and Future Work

Word sense disambiguation has long been studied as part of Artificial Intelligence and Machine Translation, but it still remains a difficult problem today. Achieving 100% accuracy has not been possible although only 40% of words in English language are polysemous. These 40% polysemous words are the ones most commonly used and hard to disambiguate.

The results of evaluation of the gloss based wsd system described in chapter 5 indicate that some more efforts are required to improve the accuracy. There are many parameters in the algorithm which vary the results, so a optimal configuration has to be formulated based on the experimental results. Right now we are not distinguishing between the glosses of different parts of speech and give them equal weight-age. A deeper study can reveal the different weight-age for glosses of words differing in parts of speech. Also knowledge bases for creating the glosses, other than WordNet also need to be investigated. Glosses often contain many noisy words which sometimes results in giving the wrong sense a higher score. Quality of glosses is also an important perspective which needs to be looked into. A system for input of glosses directly by a human user will also be useful.

Appendix A

As part of the work on WSD, fully manual disambiguation system has been developed. The user has to label the correct sense of all the ambiguous words in a document. However, monosemous words are labelled by the system itself and does not require human intervention. Figure6.1 shows a screen-shot of the system.

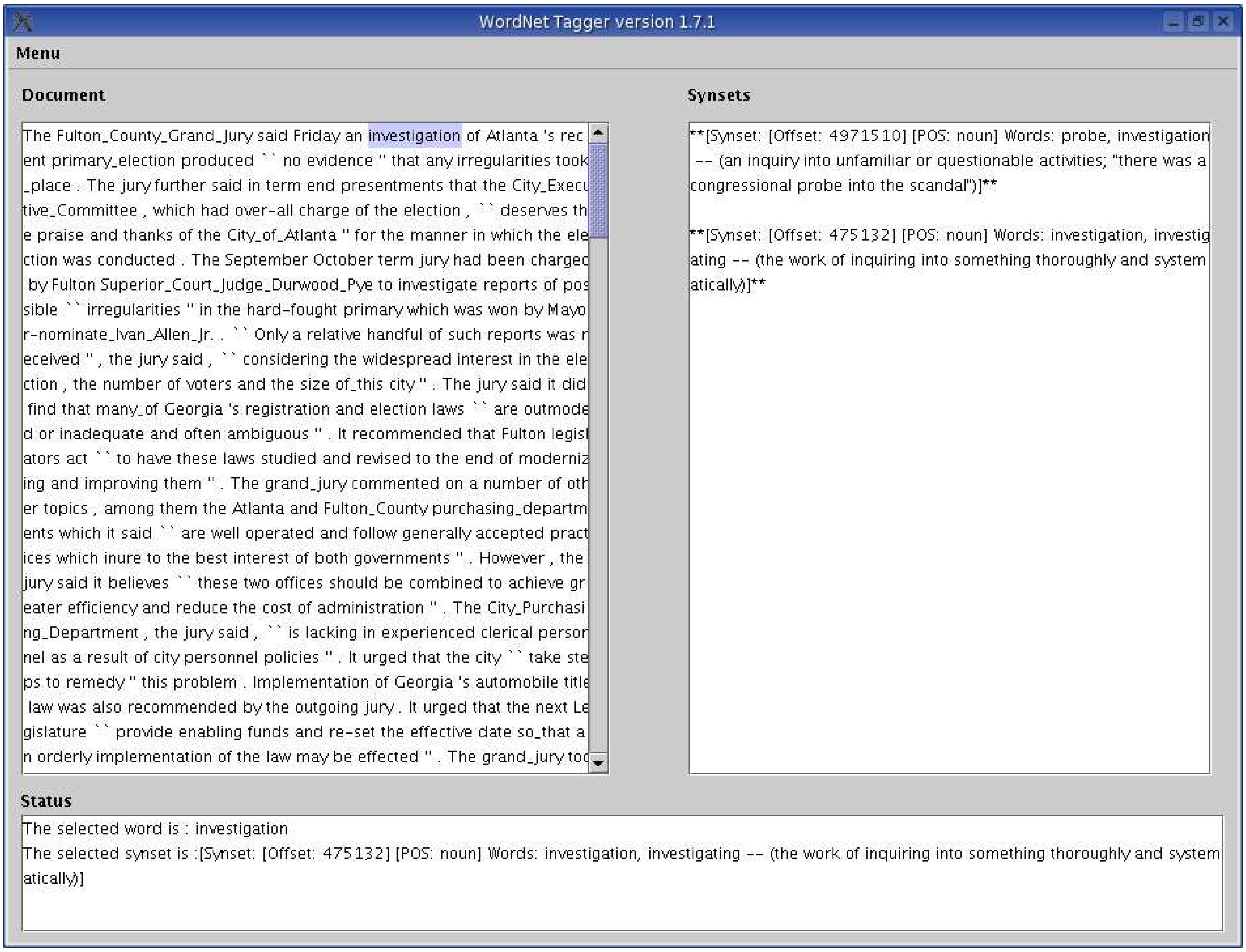

Figure 6.1: WSD system

The main screen is divided into three windows: document window, synset window and a status window. The user opens the document and it is displayed in the document window. Clicking on any word in the document window shows all the synsets (senses) of the word in the synset window. Since it is a totally manual system right now, all the senses are displayed. In an interactive WSD system if the system can disambiguate the word by itself, it will show only the disambiguated sense in the synset window. Clicking on one of the synsets in synset window will tag the corresponding word with that synset and save it in an output file. The format in which the word is tagged is as shown in figure 6.2.

The system will serve as a foundation for building interactive WSD system in which the

```
<IIT-WORD>
    <LITERAL>....</LITERAL>
    <POS>....</POS>
    <SYNSET>
        <OFFSET>........</OFFSET>
        <EXAMPLES>......</EXAMPLES>
        <GLOSS>.........</GLOSS>
    </SYSNSET>
</IIT-WORD>
```

Figure 6.2: Format for tagging a word with its synset

system will try to disambiguate as many words as possible by itself and the user will do the same thing for remaining words.

Appendix B

We have implemented several utilities as part of the WSD system. Here is a brief description about each of them.

1. **SimMeasureGUI** [URL: http://laiirs3.cse.iitb.ac.in:8080/servlet/SimMeasureGUI]
   This utility helps in finding out the similarity between two words. The gloss can be changed to any of the available types- lin, lesk etc. The words are to taken in the format of word#pos#senseno or word#pos. It compares the glosses of the two words and gives the similarity between them. If the senseno. is specified then gloss of only that sense is taken else the maximum similarity amongst all the senses of the word is reported.

2. **MainWsdGUI** [URL: http://laiirs3.cse.iitb.ac.in:8080/servlet/MainWsdGUI]
   It takes as input a pos-tagged sentence and a word and finds the sense of the word which is most related to the sentence. The gloss of the sentence is compared with gloss of each of the sense of the word.

3. **PassageWsdGUI** [URL: http://laiirs3.cse.iitb.ac.in:8080/servlet/PassageWsdGUI]
   A full passage without pos-tagging is given as input. The passage is pos-tagged and then disambiguation is done for each word in the passage.

Main Classes

- **SimMeasure** It has several useful functions. They include,

  1. **getGloss(query, glossType):** It returns the gloss of *query* corresponding to the glosstype (lin, lesk etc). The query has the format - word#pos#sensenumber.
  2. **createTFIDFVector(gloss,hash):** The idf of the words occurring in the gloss are returned in hashmap *hash*.
  3. **findSimilarity(word1, word2, typeOfGloss, typeOfSim):** Returns the similarity score between *word1* and *word2* using *typeOfGloss* gloss and *typeOfSim* (usually cosineSim).

- **MainWsd** This is the main class for disambiguation. It needs SimMeasure for several of its calls. Some important function calls are presented below. Parameters are changed through the following varaiables:

  1. **xwn:** This variable indicates whether the input is in extended WordNet form or not. If it is then the pos-tagging is not done as the words are already pos-tagged.
  2. **fullgloss:** Indicates whether the gloss of the context has to be taken or not. If set to true then the gloss of the context words will also be included.
  3. **toStem:** Indicates whether stemming is to be done or not.

  Function calls:

  1. **wsdPassage(posTagged, document, typeofgloss, simmeasure):** It is the most important method of MainWsd class. *postagged* is a boolean that indicates whether

the document is postagged or not. The passage is given as a string variable *document*. Gloss type is set through *typeOfGloss* and *simmeasure* indicates the similarity measure. The method returns a string which contains the synsets of the words with their scores.

2. **getSentenceGloss(posTagged, sentence):** The gloss of the words in the sentence is put in a hashmap. The keys are words while the values are their glosses.

Format for input/output

Apart from the web-based programs we also have a console driven interface which is implemented by the class **XwnLesk**. It was used for the experiments with Semcor and Senseval-3 task. The format is similar to the Semcor format.

```
<gloss pos=br-a01 synsetid="1">
<wsd>
<wf  pos="DT">The</wf>
<wf  pos="NNP" >Fulton_County_Grand_Jury</wf>
<wf  pos="VB" lemma="say" >said</wf>
<wf  pos="NN" lemma="friday" >Friday</wf>
<wf  pos="DT">an</wf>
<wf  pos="NN" lemma="investigation" >investigation</wf>
</wsd>
</gloss>
```

Figure 6.3: Format for input

If the sense no. is known in advance then that can also be put as an argument.

```
<wf pos="NN" lemma="friday" wnsn=1>Friday</wf>
```

The output format is,

```
<instId>br-a01.1</instId><lemma>investigation.n</lemma><sense>
investigation#n#1</sense><leskscore>0.0</leskscore><label>+1</label>

<instId>br-a01.1</instId><lemma>investigation.n</lemma><sense>
investigation#n#2</sense><leskscore>0.0</leskscore><label>-1</label>
```

Figure 6.4: Format for output

Here the label indicates the sense no which is correct. This is possible only when you have a sense-tagged corpus like Semcor.

Miscellaneous

The accuracy of the system is found using the output format as described above and a perl script called findAccuracy.pl . The script takes as parameter the part-of-speech for which we want the accuracy.

JWNL (Java WordNet API), available at http://jwordnet.sourceforge.net, was used to make WordNet calls. **IndexWord** class was used to find the no. of senses of a word and **Synset** class was used to handle the synsets of the words.

Bibliography

[1] http://clwww.essex.ac.uk/w3c/corpus_ling/content/corpora/list/private/brown/brown.html.

[2] http://www.cs.unt.edu/r̃ada/downloads.html#semcor.

[3] Word sense disambiguation. http://clr.nmsu.edu/ raz/ling5801/papers/semantic/wsd.ps.

[4] E. Agirre and G. Rigau. Word-sense disambiguation using conceptual density. In *Proceedings of the International Conference on Computational Linguistics (COLING-96)*, 1996.

[5] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Proceedings of the Royal Statistical Society*, pages 1–38, 1976.

[6] G. A. Miller, R. Beckwith, C. Fellbaum, D. Gross, and K. J. Miller. Introduction to wordnet: an on-line lexical database. *International Journal of Lexicography 3 (4)*, pages 235 – 244, 1990.

[7] J. Pearl. *Probabilistic Reasoning in Intelligent Systems : Networks of Plausible Inference.* Morgan Kaufmann, 1988.

[8] G. Ramakrishnan and P. Bhattacharya. Text representation with wordnet synsets: A soft sense disambiguation approach. In *Proceedings of 8th International Conference on Applications of Natural Language to Information Systems*, 2003.

[9] G. Ramakrishnan, Prithviraj B.P., A. Deepa, P. Bhattacharya, and S. Chakrabarti. Soft Word Sense Disambiguation. In *Proceedings of the Second International WordNet Conference—GWC 2004*, pages 291–298, Brno, Czech Republic.

[10] D. Yarowsky. Word-sense disambiguation using statistical models of roget's categories trained on large corpora. In *Proceedings of the 14th International Conference on Computational Linguistics (COLING-92)*, pages 454–460, Nantes, France, 1992.