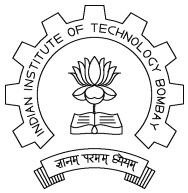
Word Sense Disambiguation

First Stage Report

Submitted in partial fulfilment of the requirements
of the degree of
Master of Technology
by
Esha Palta
(Roll No. 05329017)

Under the guidance of
Prof. Om Damani
Kanwal Rekhi School of Information Technology



Kanwal Rekhi School of Information Technology
Indian Institute of Technology, Powai, Mumbai
2006-2007

Abstract

Word sense disambiguation (WSD) is the task of selecting the appropriate senses of a word in a given context. It is essence of communication in a natural language. It is motivated by its use in many crucial applications such as Information retrieval, Information extraction, Machine Translation, Part-of-Speech tagging, etc. Various issues like scalability, ambiguity, diversity (of languages) and evaluation pose challenges to WSD solutions. The aim of this project is to develop a WSD technique which can handle all these issues with better accuracy and performance. This report presents our preliminary work towards solving the problem.

1 Introduction

"When I use a word," Humpty Dumpty said,...
"it means just what I choose it to mean – neither more nor less."

-Lewis Carroll (1875) from [1]

Words can have different senses. Some words have multiple meanings. This is called *Polysemy*. For example: bank can be a financial institute or a river shore. Sometimes two completely different word are spelled the same. For example: *Can*, can be used as model verb: You *can* do it, or as container: She brought a *can* of soda. This is called *Homonymy*. Distinction between polysemy and homonymy is not always clear. Word sense disambiguation (WSD) is the problem of determining in which sense a word having a number of distinct senses is used in a given sentence. Take another example, consider the word "bass", with two distinct senses:

1. a type of fish
2. tones of low frequency

and the sentences "The bass part of the song is very moving" and "I went fishing for some sea bass". To a human it is obvious the first sentence is using the word "bass" in sense 2 above, and in the second sentence it is being used in sense 1. But although this seems obvious to a human, developing algorithms to replicate this human ability is a difficult task.

One problem with word sense disambiguation is deciding what are the senses. In cases like the word "bass" above, at least some senses are obviously different. In other cases, however, the different senses can be closely related (one meaning being a metaphorical extension of another), and in such cases division of words into senses becomes much more difficult. Consulting different dictionaries will find many different divisions of words into senses. One solution some researchers have used is to choose a particular dictionary, and just use its set of senses. There are two main approaches to WSD – deep approaches and shallow approaches.

Deep approaches presume access to a comprehensive body of world knowledge. Knowledge such as "you can go fishing for a type of fish, but not for low frequency sounds" and "songs have low frequency sounds as parts, but not types of fish" is then used to determine in which sense the word is used. These approaches are not very successful in practice, mainly because we don't have access to such a body of knowledge, except in very limited domains. But if such knowledge did exist, they would be much better than the shallow approaches.

Shallow approaches don't try to understand the text. They just consider the surrounding words, using information like "if 'bass' has words 'sea' or 'fishing' nearby, it probably is in the fish sense; if 'bass' has the words 'music' or 'song' nearby, it is probably in the music sense." These rules can be automatically derived by the computer, using a training corpus of words tagged with their word senses. This approach, while theoretically not as powerful as deep approaches, gives superior results in practice, due to our limited world knowledge. It can, though, be confused by sentences like "The dog barked at the tree."

This report highlights main issues in word sense disambiguation, various approaches and solutions proposed till date.

2 Motivation

Word sense disambiguation a task of removing the ambiguity of word in context, is important for many NLP applications such as:

- Information Retrieval: As proposed by [2] WSD helps in improving term indexing in information retrieval. [2] has proved that word senses improve retrieval performance if the senses are included as index terms. Thus, documents should not be ranked based on words alone, the documents should be ranked based on word senses, or based on a combination of word senses and words. For example: Using different indexes for keyword "Java" as "programming language", as "type of coffee", and as "location" will improve accuracy of an IR system. Apart from indexing, WSD also helps in query expansion. Short queries are expanded using words that belong to same syn-sets. Retrieval using expanded queries gives better results than original queries. Thus, WSD is crucial for improving accuracy of IR as it eliminates irrelevant hits.

- Machine Translation: WSD is important for Machine translations. It helps in better understanding of source language and generation of sentences in target language. It also affects lexical choice depending upon the usage context.

- Speech Processing and Part of Speech tagging[1]: Speech recognition i.e, when

[1]POS is the process of marking up the words in a text as corresponding to a particular part of speech, based on both its definition, as well as its context.

processing homophones words which are spelled differently but pronounced the same way. For example: “base” and “bass” or “sealing” and “ceiling”.

- Text Processing: Text to Speech translation i.e, when words are pronounced in more than one way depending on their meaning. For example: “lead” can be “in front of” or “type of metal”.

3 Problem Definition

Word sense disambiguation (WSD) involves the association of a given word in a text or discourse with a definition or meaning which is distinguishable from other meanings potentially attributable to that word. The task therefore necessarily involves two steps according to Ide and Veronis (1998). The first step is to determine all the different senses for every word relevant to the text or discourse under consideration, i.e., to choose a sense inventory, e.g., from the lists of senses in everyday dictionaries, from the synonyms in a thesaurus, or from the translations in a translation dictionary.

The second step involves a means to assign the appropriate sense to each occurrence of a word in context. All disambiguation work involves matching the context of an instance of the word to be disambiguated either with information from external knowledge sources or with contexts of previously disambiguated instances of the word. For both of these sources we need preprocessing or knowledge-extraction procedures representing the information as context features. For some disambiguation tasks, there are already well-known procedures such as morpho-syntactic disambiguation and therefore WSD has largely focused on distinguishing senses among homographs belonging to the same syntactic category.

Finally a third step is also involved: the computer needs to learn how to associate a word sense with a word in context using either machine learning or manual creation of rules or metrics. Main focus of this report is the use of machine learning approaches for WSD. In these approaches, systems are trained to perform the task of word sense disambiguation. In these approaches first a classifier is learned from the training examples, which is later used to assign senses to unseen examples.

4 Literature Survey

A survey of learning methodologies that have been used for WSD is presented in Section 4.1. Following these the studies of works based on some of these methodologies is presented in Section 4.2.

4.1 Learning Methodologies

WSD methods can be classified into two types: Machine learning approaches and Dictionary based approaches. WSD that use information gathered from training

on a corpus is based on machine learning approaches. WSD that make use of the information provided by Machine readable dictionaries (MRD) is based on dictionary based approaches. Section 4.1.1 discuss basic Machine learning approaches and section 4.1.2 discusses basic Dictionary based approaches.

4.1.1 Machine Learning approaches

Machine learning approaches systems are trained to perform task of word sense disambiguation. In this method a classifier is learned which is then used to assign senses to unseen examples. In these approaches, the initial input consist of the word to be disambiguated (**target**) word, along with text in which it is embedded which is called as context. This initial input is processed using part-of-speech tagging or any morphological processing. After this initial processing, fixed set of linguistic features are extracted relevant to learning task. These features are of two classes: collocation and co-occurrence features. *Collocation* features encode information about words of specific positions that are located to left or right of target word. Typical features include the word, the root form of word, and the word's part-of-speech. Consider an example:
An electric guitar and **bass** player stand off to one side, not really part of scene.
Here we need to disambiguate word bass, so its our target word. Collocation feature vector considering two words to right and two words to left of target words is:

$$guitar, NN1^{2}, and, CJC^{3}, player, NN1, stand, VVB$$

Co-occurrence features consist of data about neighboring words. In this approach words themselves serve as features. The value of feature is the number of times the word occurs in the region surrounding the target word. The region is often a fixed window with target word as center. For the earlier example, a co-occurrence vector consisting of 12 most frequent words from a collection of *bass* sentences drawn from WSJ corpus has following features: *fishing, big, sound, player, fly, rod, pound, double, runs, playing, guitar, band.* Using these words as features with a window size of 10, earlier example would be represented as following vector:

$$0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0$$

Most of the approaches to sense disambiguation make use of both of these features.

- Supervised Techniques: Words can be labeled with their senses
  - She pays 3% interest/INTEREST-MONEY on the loan.
  - He showed a lot of interest/INTEREST-CURIOSITY in the painting.

  Supervised approaches are therefore similar to tagging:

[2] noun
[3] conjunction

– given a corpus tagged with senses
– define features that indicate one sense over another
– learn a model that predicts the correct sense given the features

In supervised approaches, a sense disambiguation system is learned from a representative set of labeled instances drawn from same distribution as test set to be used. Input instances to these approaches are feature encoded along with their appropriate labels. The output of the system is a classifier system capable of assigning labels to new feature encoded inputs. Following are some of supervised techniques:

– Naive Bayes Classifier: It is based on the premise that choosing the best sense for input vector amounts to choosing the most probable sense.

$$\widehat{s} = argmax_{s \in S} P(s|V) \tag{1}$$

S denotes set of senses appropriate for the target associated with this vector, s denotes each of possible sense in S, and V stands for the vector representation of the input context. Now applying bayes rule we get,

$$\widehat{s} = argmax_{s \in S} \frac{P(s|V)P(s)}{P(V)} \tag{2}$$

Considering the earlier "bass" example, individual statistics needed for this example might include the probability of the word "player" coming to the immediate right of a use of each of bass senses.

– Decision Lists: These classifiers are equivalent to simple case statements in most programming languages. In decision list classifier a sequence of tests is applied to each vector encoded input. If test succeeds then the sense associated with that test is returned. If test fails then next test in the sequence is applied. This continues until the end of list, where a default test simply returns majority sense. Decision Lists have,
  * two senses per word
  * rules of the form: $collocation \longrightarrow sense$
  * example: manufacturing $plant \longrightarrow PLANT - FACTORY$
  * rules are ordered, most reliable rules first
  * when classifying a test example, step through the list, make decision on first rule that applies

  Learning: rules are ordered by,

$$log(\frac{P(sense_A|collocation_i)}{P(sense_B|collocation_i)}) \tag{3}$$

- Semi-Supervised Techniques: The semi-supervised or minimally supervised methods are gaining popularity because of their ability to get by with only a small amount of annotated reference data while often outperforming totally unsupervised methods on large data sets. There are a host of diverse methods and approaches, which learn important characteristics from auxiliary data and cluster or annotate data using the acquired information.

- Unsupervised Techniques: Unsupervised approaches to sense disambiguation eschew the use of sense tagged data of any kind during the training. In this technique, feature vector representations of unlabeled instances are taken as input and are then grouped into clusters according to a similarity metric. These clusters are then labeled by hand with known word senses. Main disadvantage is that senses are not well defined.

4.1.2 Dictionary Based Approaches

In this style of approach the dictionary provides both the means of constructing a sense tagger and target senses to be used. Attempts to perform large scale disambiguation has lead to the use of Machine Readable Dictionaries (MRD). In this approach, all the senses of a word to be disambiguated are retrieved from the dictionary. Each of these senses is then compared to the dictionary definitions of all the remaining words in context. The sense with highest overlap with these context words is chosen as the correct sense. For example: consider the phrase *pine cone* for selecting the correct sense of word *cone* and following definitions for *pine* and *cone*:
pine:

1. kinds of evergreen tree with needle-shaped leaves
2. waste away through sorrow or illness

cone:

1. solid body which narrows to a point
2. something of this shape whether solid or hollow
3. fruit of certain evergreen trees

In this example, Lesk's method would select **cone**[3] as the correct sense since two of the words in its entry, evergreen and tree, overlap with words in the entry for *pine*. A major drawback of Dictionary based approaches is the problem of scale.

4.2 Related Work

This section gives brief description of the various solutions proposed for word sense disambiguation problem. Most of the algorithms makes use of the specific properties of word co-occurrence graphs. Unlike earlier dictionary-free methods based on

word vectors, it can isolate highly infrequent uses (as rare as 1% of all occurrences) by detecting "hubs" and high-density components in the co-occurrence graphs.

4.2.1 HyperLex

The HyperLex algorithm presented in [3] (Veronis, 2004) is entirely corpus-based. It builds a co occurrence graph for all pairs of words concurring in the context of the target word. Veronis shows that this kind of graph fulfills the properties of small world graphs, and thus possesses highly connected components (hubs) in the graph. These hubs eventually identify the main word uses (senses) of the target word, and can be used to perform word sense disambiguation. These hubs are used as a representation of the senses induced by the system. The basic steps for implementation of HyperLex are: We first build the co occurrence graph, then we select the hubs that are going to represent the senses using two different strategies inspired by HyperLex. We are then ready to use the induced senses to do word sense disambiguation. Consider as an example a french word *barrage*, on which a query may return pages on dams, play-offs, barriers, roadblocks, police cordons, barricades, etc. Main steps of algorithm are explained using this example as follows:

- Building Co-occurrences graphs: For each word to be disambiguated, a text corpus is collected, consisting of the paragraphs where the word occurs. From this corpus, a co-occurrence graph for the target word is built. Vertexes in the graph correspond to words[4] in the text (except the target word itself). Two words appearing in the same paragraph are said to coccus, and are connected with edges. Each edge is assigned a weight which measures the relative frequency of the two words co-occurring. [3] has proposed w_{ij} to be the weight of the edge[5] connecting nodes i and j, then $w_{ij} = 1 - max[P(i|j), P(j|i)]$; where $P(i|j) = \frac{freq_{ij}}{freq_j}$ and $P(j|i) = \frac{freq_{ij}}{freq_i}$.

  The weight of an edge measures how tightly connected the two words are. Words which always occur together receive a weight of 0. Words rarely co occurring receive weights close to 1. Consider the following context with target word as *barrage*:
  Outre la ***production*** *d'****électricité****, le* ***BARRAGE*** *permettra de réguler le cours du fleuve...* (In addition to the **production** of **electricity**, the **DAM** will help regulate the river flow ...)

  Co-occurrence graph of target word in figure 1 *barrage* shows that nodes corresponding to *production* and *électricité* are connected to each other as they occur together in this context. The basic assumption underlying the method proposed here is that the different uses of a target word form highly interconnected "bundles" in a small world of co-occurrences, or in terms of graph theory, high density components. Accordingly, *barrage* (in the sense

[4] only nouns.

[5] The co-occurrence graph is undirected, i.e. $w_{ij} = w_{ji}$

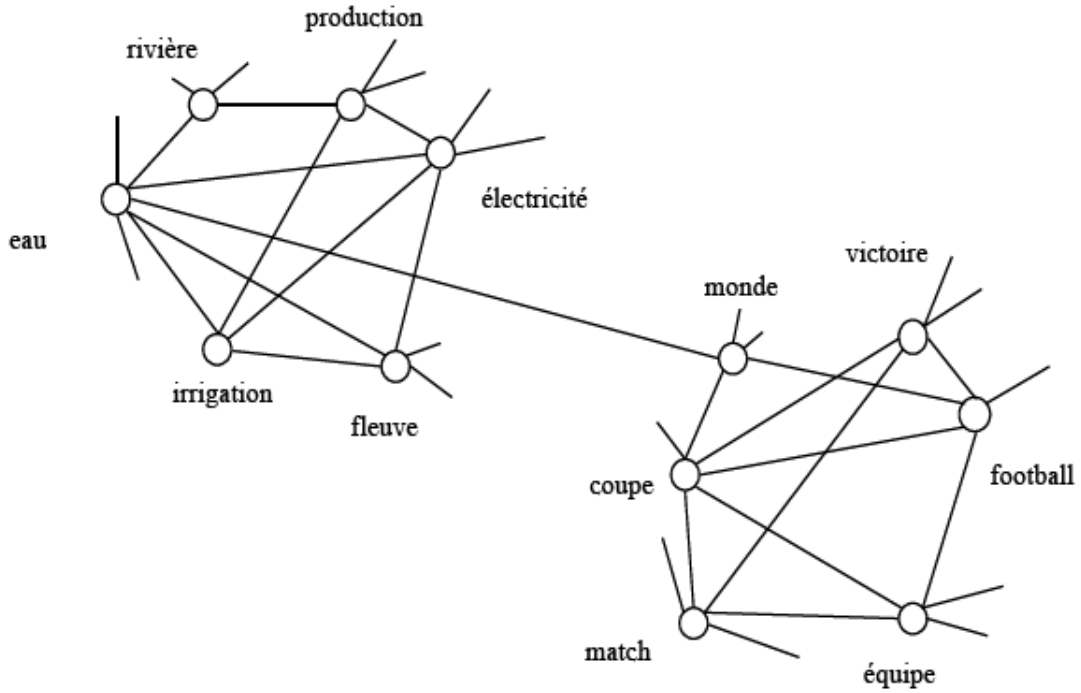


Figure 1: Graph of the cooccurrents of the French word barrage from [3]

of a hydraulic dam) must co-occur frequently with eau, ouvrage, riviére, crue, irrigation, production, électricité (water, work, river, flood, irrigation, production, electricity), etc., and these words themselves are likely to be interconnected as shown in figure 1. Similarly, in the play-off use, barrage must cooccur frequently with match, quipe, coupe, monde, football, victoire (match, team, cup, world, soccer, victory), etc., which again are highly interconnected.

- Selecting Hubs: Once the co-occurrence graph is built, Veronis proposes a simple iterative algorithm to obtain its hubs. At each step, the algorithm finds the vertex with highest relative frequency[6] in the graph, and, if it meets some criteria, it is selected as a hub. For example, for the most frequent use of *barrage* (hydraulic dam), the root hub is the word eau (water). After a vertex is selected to be a hub, its neighbors are no longer eligible as hub candidates. At any time, if the next vertex candidate has a relative frequency below a certain threshold, the algorithm stops. In *barrage* example once again, the four components can be characterized as follows: EAU,

[6]In co-occurrence graphs, the relative frequency of a vertex and its degree are linearly related, and it is therefore possible to avoid the costly computation of the degree.

ROUTIER, FRONTIERE, MATCH.

- Using Hubs for WSD: Once the hubs that represent the senses of the word are selected (following any of the methods presented in the last section), each of them is linked to the target word with edges weighting 0, and the Minimum Spanning Tree (MST) of the whole graph is calculated and stored. Final MST for our example is figure 2. The minimum spanning tree can be used

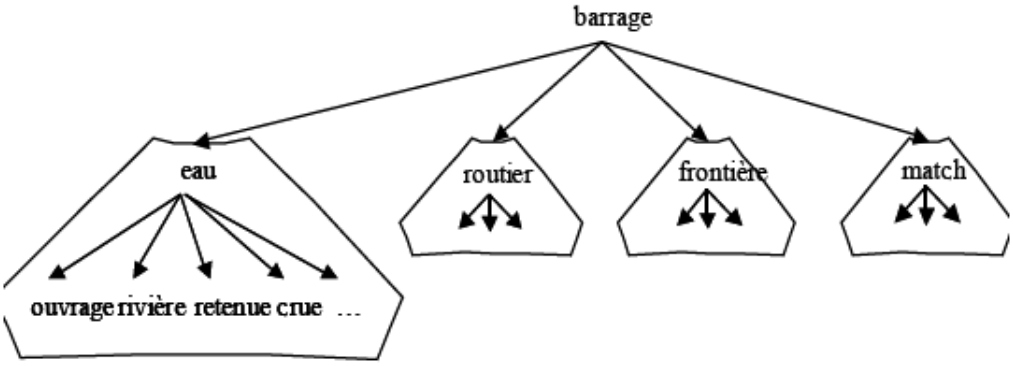


Figure 2: Minimum spanning tree and high-density components from [3]

to easily construct a disambiguation for tagging target word occurrences in the corpus. Each tree node v is assigned a score vector s with as many dimensions as there are components:

$$s_i = \frac{1}{1 + d(h_i, v)} \quad \texttt{if v belongs to component i} \tag{4}$$

$$s_i = 0 \quad \texttt{otherwise.} \tag{5}$$

This formula (from [3]) assigns a score of 1 to root hubs, whose distance from themselves is 0. The score gradually approaches 0 as the nodes move away from their root hub. For example, pluie (rain) belongs to the component EAU (water) and d(eau, pluie) = 0.82; its score vector is (0.55 0 0 0). Likewise, saison (season) belongs to the component MATCH and d(match, saison) = 1.54; its score vector is (0 0 0 0.39).

For a given occurrence of the target word, the score vectors of all the words in the context are added, and the hub that receives the maximum score is chosen.

4.2.2 PageRank

PageRank algorithm (Brin and Page, 1998) is similar in all aspects to HyperLex algorithm explained in section 4.2.1 except in process of finding hubs in the co

occurrence graph. PageRank is an iterative algorithm that ranks all the vertexes according to their relative importance within the graph following a random-walk model. In this model, a link between vertices's v1 and v2 means that v1 recommends v2. The more vertices's recommend v2, the higher the rank of v2 will be. Furthermore, the rank of a vertex depends not only on how many vertices point to it, but on the rank of these vertices as well. Although PageRank was initially designed to work with directed graphs, and with no weights in links, the algorithm can be easily extended to model undirected graphs whose edges are weighted. Specifically, let G = (V,E) be an undirected graph with the set of vertices V and set of edges E. For a given vertex v_i, let $In(v_i)$ be the set of vertices pointing to it5. The rank (from [4]) of v_i is defined as:

$$P(v_i) = (1-d) + d \sum_{j \in ln(v_i)} \frac{w_{ji}}{\sum_{k \in ln(v_j)} w_{jk}} P(v_j) \tag{6}$$

where w_{ij} is the weight of the link between vertices v_i and v_j, and $0 \leq d \leq 1$. d is called the damping factor and models the probability of a web surfer standing at a vertex to follow a link from this vertex (probability d) or to jump to a random vertex in the graph (probability 1 - d). The factor is usually set at 0:85. The algorithm initializes the ranks of the vertices with a fixed value and iterates until convergence below a given threshold is achieved, or, more typically, until a fixed number of iterations are executed. Note that the convergence of the algorithms doesn't depend on the initial value of the ranks. After running the algorithm, the vertices of the graph are ordered in decreasing order according to its rank, and a number of them are chosen as the main hubs of the word.

The performance of PageRank is statistically the same as that of HyperLex, with the advantage of PageRank of using less parameters.

4.2.3 Random Walk Algorithm

Many natural language processing tasks consist of labeling sequences of words with linguistic annotations, e.g. word sense disambiguation, part-of-speech tagging, named entity recognition, and others. A graph-based sequence data labeling algorithm is presented in [5] as solution for such natural language annotation tasks. The algorithm simultaneously annotates all the words in a sequence by exploiting relations identified among word labels, using random walks on graphs encoding label dependencies.

The basic idea implemented by an iterative graph based ranking algorithm is that of "voting" or "recommendation". When one vertex links to another one, it is basically casting a vote for that other vertex. The higher the number of votes that are cast for a vertex, the higher the importance of the vertex. Given a graph G = (V, E), let $In(V_a)$ be set of incoming vertices and $Out(V_a)$ be the set of vertices the vertex V_a points to. Page Rank (from [5]) is given by:

$$P(V_a) = 1 - d + d * \sum_{V_b \in In(V_a)} \frac{P(V_b)}{|Out(V_b)|} \tag{7}$$

where d is a parameter that is set between 0 and 1.
Algorithm for Sequence Data Labeling has following steps:

- Construction of label dependencies graph
- Label scoring using graph-based ranking algorithms
- Label assignment

The **church bells** no longer **rung** on **Sundays**.

church
1: one of the groups of Christians who have their own beliefs and forms of worship
2: a place for public (especially Christian) worship
3: a service conducted in a church

bell
1: a hollow device made of metal that makes a ringing sound when struck
2: a push button at an outer door that gives a ringing or buzzing signal when pushed
3: the sound of a bell

ring
1: make a ringing sound
2: ring or echo with sound
3: make (bells) ring, often for the purposes of musical edification

Sunday
1: first day of the week; observed as a day of rest and worship by most Christians

Figure 3: Senses of words from WordNet (from [5])

Consider an example for explaining this algorithm. Consider the task of assigning senses to the words in the text "The church bells no longer rung on Sundays." Assume at most three senses for each word as shown in Figure 3. All word senses are added as vertices in the label graph, and weighted edges are drawn as dependencies among word senses, derived using the definition-based similarity measure. The resulting label graph is an undirected weighted graph, as shown in Figure 4. After running the ranking algorithm, scores[7] are identified for each word-sense in the graph, indicated between brackets next to each node. Selecting for each word the sense with the largest score results in the following sense assignment: The church#2 bells#1 no longer rung#3 on Sundays#1, which is correct.

[7]Edge weights and vertex scores are computed in same way as done for HyperLex algorithm explained in section 4.2.1.

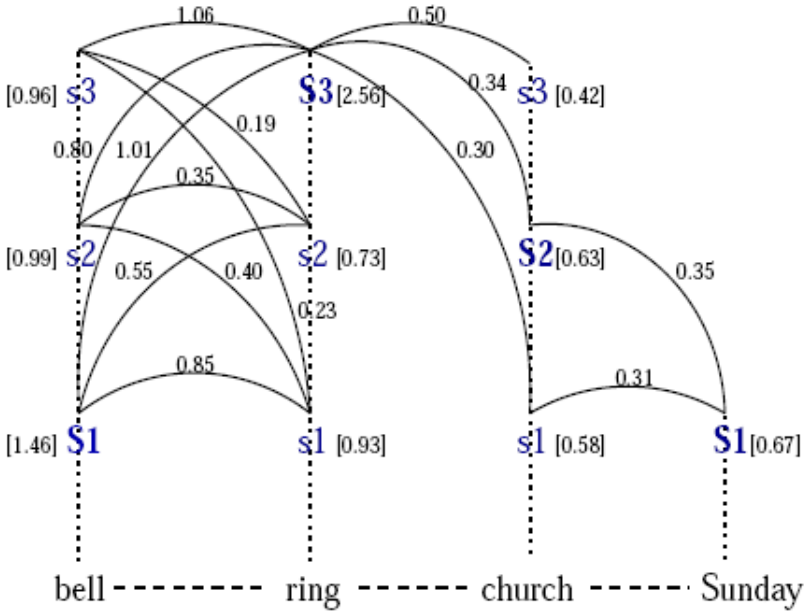


Figure 4: The label graph for assigning senses to words in the sentence The church bells no longer rung on Sundays.(from [5])

4.2.4 Non-ambiguous words

Many of the supervised techniques require annotated data as input. [6] has addressed this problem of obtaining annotated data required by some of the supervised algorithms. This method takes as input the raw textual corpus, generate lexical knowledge from non-ambiguous words via classes of equivalence[8] and enables automatic generation of annotated corpora. An ambiguous word is one which has more than one possible tag, for example "work" can be either a noun or verb, while a non-ambiguous word carry only one tag.

The basic idea is that for an ambiguous word W, an attempt is made to identify one or more non-ambiguous words W' in the same class of equivalence, so that W' can be annotated in the automatic fashion. Next, lexical knowledge is induced from the non-ambiguous words W' to ambiguous words W using classes of equivalence. The knowledge step is performed using learning mechanism, where the automatically partially tagged corpus is used for training to annotate new raw texts including instances of the ambiguous word W.

For an illustration, consider the process of assigning a part of speech label to the word "work" which can have label NN (noun) or VB (verb). We identify in the corpus all instances of words that were already annotated with one of these

[8]Equivalence class can be represented by set of words that have same functionality (e.g. noun or same meaning)

two labels. These instances constitute training examples, annotated with one of the classes NN or VB. A classifier is then trained on these examples using features extracted from these instances, and used to automatically assign a label to the current ambiguous word "work".

This algorithm basically generate annotated data from raw text corpus and uses the idea that a classifier learned from non-ambiguous words can be used to annotate ambiguous words correctly.

4.2.5 Word-word dependency approach

The approach presented in [7] takes advantage of the sentence context. The words are paired and an attempt is made to disambiguate one word within the context of other word. This is done by searching on Internet with queries formed using different senses of one word, while keeping another one fixed. The senses are simply ranked by the order provided by the number of hits. Thus all the words are processed and senses are ranked. The next step is to refine the ordering of senses using a *semantic density*. This is measured by the number of common words that are within a semantic distance of two or more words.

This work has addressed a problem of determining the exact sense of words when present in combination. Except from determining the exact senses it also provide the ranking of different senses which is useful for many applications such as Information extraction.

5 Future Work

We've surveyed various supervised, unsupervised, dictionary based approaches as a solution for WSD. We also come to know about many related problems such as annotating the untagged raw corpus, exact sense of words when present in combinations such as noun-noun, verb-noun, verb-verb and ranking different senses. After survey or initial overview we would like to do the following:

- **We would like to analyse and evaluate existing** supervised, unsupervised and dictionary based approaches to know about the strengths and shortcomings of the existing systems.
- **Improvement in earlier proposed solutions, if possible.**
- **We would also like to investigate deep approaches in detail and integrate different kinds of information**, i.e. integeration of the local or syntactic features (used successfully by supervised systems) alongwith heterogeneous information from knowledge bases.
- **Explore other approaches.**

The final aim of this project is to develop a solution for word sense disambiguation problem that is fast, scalable, efficient in terms of performance and accuracy and more useful for various applications such as Information retrieval, Machine Translation, etc.

References

[1] Krister LINDEN. Word sense discovery and disambiguation.

[2] Robert Krovetz and W. Bruce Croft. Lexical ambiguity and information retrieval. *Information Systems*, 10(2):115–141, 1992.

[3] Jean Veronis. Hyperlex: lexical cartography for information retrieval. *Computer Speech & Language*, 18(3):223–252, 2004.

[4] Oier Lopez de Lacalle Eneko Agirre, David Martinez and Aitor Soroa. Two graph-based algorithms for state -of-the-art wsd. NLP Group, University of the Basque Country, Donostia, Basque Contry, 2006.

[5] Rada Mihalcea. Unsupervised large-vocabulary word sense disambiguation with graph-based algorithms for sequence data labeling. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 411–418, Vancouver, British Columbia, Canada, October 2005. Association for Computational Linguistics.

[6] M. Rada. The role of non-ambiguous words in natural language disambiguation, 2003.

[7] R. Mihalcea and D. Moldovan. A method for word sense disambiguation of unrestricted text, 1999.

[8] G. Ramakrishnan and B. Prithviraj. Soft word sense disambiguation, 2004.

[9] Amruta Purandare and Ted Pedersen. Word sense discrimination by clustering contexts in vector and similarity spaces.

[10] R. Philip and D. Yarowsky. Distinguishing systems and distinguishing senses: new evaluation methods for word sense disambiguation, 1999.

[11] Prabhakar Pande Lakshmi Kashyap Manish Sinha, Mahesh Kumar and Pushpak Bhattacharyya. Hindi word sense disambiguation. In *International Symposium on Machine Translation*, Natural Language Processing and Translation Support Systems, Delhi, India, November 2004.