CHAPTER 1
INTRODUCTION

1.1 Word Sense Disambiguation

Natural language is inherently ambiguous with many words having different meanings depending on the context where they occur. For example, the word bank can mean “financial institution” or “river side”. Similarly, the word “bat” can mean “a nocturnal creature” or “a piece of sports equipment”. The task of finding the correct meaning (sense) of a word in a particular context in a computational manner is known as Word Sense Disambiguation (WSD) [5, 6]. Determining the appropriate meaning of ambiguous words in context is intuitive to humans, but for machines simulating the process has proved to be one of the most difficult task in Natural Language Processing (NLP).

The context of the word to be disambiguated is the information contained within the text or discourse in which the word appears. Generic WSD can be either Lexical Sample (targeted WSD) or All-words WSD. In lexical sample, a system is required to disambiguate a restricted set of target words usually occurring one per sentence, whereas in All-words WSD, systems are expected to disambiguate all open-class words in a text (i.e.; nouns, verbs, adjectives, and adverbs).

The meanings or senses in NLP, can be classified as coarse-grained or fine-grained. At a course grain a word has a small number of senses that are clearly different and completely unrelated to each other, also called homographs [6]. For example, bank as “financial institution” and bank as “river side” are coarse-grained senses. Fine-grained senses of a word contain interrelated senses, which are obtained by breaking down the coarse-grained senses. For example, bank as “financial institution” is splitted into following senses in WordNet (a machine readable dictionary): the company or institution, the building, a supply or stock held in reserve for future use, the funds held by a gambling house, a money box. Ide and Wilks [6] argue that machines and humans are only able to reliably disambiguate coarse-grained senses, but researches have shown that using fine-grained repository such as WordNet [7], machines are capable of disambiguating fine grained senses to a reasonable degree.

1.2 Applications of WSD

WSD is a task considered essential in many NLP tasks. A number of real world applications which might benefit from WSD are:

1.2.1 Machine Translation (MT)

MT was the original problem that gave rise to WSD. Since different senses of a word have different translations across languages, all MT systems implement some sort of WSD. Most MT systems do not use an explicit WSD system because of the structure of the MT algorithms, still there is scope for a change in the algorithms so that WSD can be used explicitly. Carpuat and Wu [17], and Chan [18] showed that WSD can help improve machine translation.

1.2.2 Information Retrieval (IR)

IR often requires that ambiguities be resolved before certain queries can be performed. Early experiments suggested that reliable IR would require at least 90% disambiguation accuracy for explicit WSD to be of benefit [19]. Recently, WSD has been shown to improve cross-lingual IR [20] and document classification [21]. IR related applications like news recommendation, alerting, topic tracking and automatic advertisement placement can also benefit from WSD system.

1.2.3 Information Extraction (IE) and Text Mining

WSD is required for the accurate analysis of text in many applications [7]. Bioinformatics research requires the relationships between genes and gene products to be catalogued from the vast scientific literature. The semantic web requires automatic annotation of documents. Named-entity classification, acronym expansion (MG as magnesium or milligram) are other applications that could be benefited from WSD.

1.2.4 Lexicography

Lexicography is the professional writing of dictionaries. WSD and lexicography can work in a loop. WSD provide empirical sense groupings and statistically significant contextual indicators of sense to lexicographers, who provide better sense inventories and sense annotated corpora to WSD.

1.2.5 Content Analysis

The analysis of the general content of a text in terms of its ideas, themes, etc can be benefited from the application of WSD. The classification of blogs [22] and semantic social network analysis are active research areas in content analysis.

1.2.6 Question Answering (QA)

The aim of QA is to find answers in open-domain natural language text. Explicit sense disambiguation is found useful in QA [23].

1.3 Problem Definition

Given a text T as a sequence of words (w_1, w_2, …, w_n). The generic problem of WSD is to identify a mapping A from words to senses, such that $A(i) \subseteq \text{Senses}_D(w_i)$, where $\text{Senses}_D(w_i)$ is the set of senses encoded in a dictionary D for word w_i and A(i) is the subset of the senses of w_i which are appropriate in the context T. The mapping can assign more than one sense to each word $w_i \in T$, but typically only the most appropriate sense is selected, that is, $| A(i) | = 1$.

This dissertation try to solve the problem of WSD through graph based approach, by constructing a semantic network containing related senses of words to be disambiguated as vertices where an edge between vertices represent the semantic relation. The weight on the edge between two sense vertices represents the measure of semantic relatedness between two vertices. Disambiguation is done by finding out the most important sense vertex for each word by applying graph centrality or node ranking algorithms (Degree centrality, PageRank, etc). For each word, the sense which is considered most important by the algorithm is considered the most appropriate sense. The approach presented does not require any manually labeled data for training algorithms as in the case of supervised approaches. But only require a knowledge source or reference dictionary like WordNet that provide explicit relations between word senses. Hence, the approach is considered unsupervised.

1.4 Research Objectives

Following are the objectives of the research.

1. Study the problem of WSD and its application.
2. Study the different graph based unsupervised WSD methods.
3. Experiment and comparison of graph based centrality algorithms on semantic network of senses constructed from WordNet for WSD.

CHAPTER 2
LITERATURE REVIEW

2.1 A Brief History of WSD

WSD is one of the oldest problems in computational linguistics. WSD was first formulated as a separate task in Machine Translation (MT) by Weaver [26] in 1940s. From the very beginning WSD was considered a difficult problem. In 1960s, most of MT was abandoned due to its acknowledged hardness. In 1970s, the problem of WSD was initiated with AI approaches on complete natural language understanding. Wilks's preference semantics [27] was one of the first systems to explicitly account for WSD. Because of the lack of large amounts of machine readable knowledge, generalizing the results were still difficult.

WSD reached a turning point in the 1980s with the release of large scale lexical resources, which enabled automatic methods for knowledge extraction. Lesk [28] in 1986, with a simple and seminal algorithm used the overlap of word sense definitions in the Oxford Advanced Learner's Dictionary of Current English (OALD) to resolve word senses. Dictionary based WSD had begun and the relationship of WSD to lexicography became explicit.

The 1990s saw three major developments, WordNet became available, the statistical revolution in NLP swept through, and SENSEVAL (later SEMEVAL, www.senseval.org) began.

WordNet pushed research forward because it was both computationally accessible and hierarchically organized into word senses called synsets. Today, English Wordnet together with WordNet for other languages is the most used general sense inventory in WSD research.

The mainstream approach in WSD has been supervised learning where systems are trained on manually tagged corpora. In 1990s, many researchers successfully applied statistical and machine learning method to the WSD problem.

SENSEVAL made it possible for researchers to compare different systems with each other because of fixed set of test words, annotators, sense inventories, and corpora.

In SENSEVAL-3 (2004), a conclusion was reached that WSD in itself has reached a performance plateau, and no significant leap in the results obtained previously is possible. So, there is need of new directions in WSD research. Use of parallel bilingual corpora for WSD and unsupervised WSD are emerging directions in WSD research.

2.2 Knowledge Sources for WSD

Knowledge is a fundamental component of WSD. Knowledge sources provide data which are essential to associate senses with words. They can be corpora of texts, machine-readable dictionaries, thesauri, glossaries, ontologies, etc. Commonly used knowledge sources are described below:

2.2.1 Structured Resources

2.2.1.1 Thesauri

These provide information about relationships between words, like synonymy, antonymy, etc. The most widely used thesaurus in the field of WSD is Roget's International Thesaurus. Some researchers also use the Macquarie Thesaurus, which encodes more than 200,000 synonyms.

2.2.1.2 Machine-readable Dictionaries (MRDs)

These have become a popular source of knowledge for natural language processing. Examples are: Collins English Dictionary, Oxford Advanced Learner's Dictionary of Current English, Oxford Dictionary of English, Longman Dictionary of Contemporary English (LDOCE), WordNet, etc. Presently, WordNet is the most utilized resource for WSD in English, and it contains rich semantic network of concepts.

2.2.1.3 Ontologies

These are specifications of conceptualizations of specific domains of interest, including a taxonomy and a set of semantic relations. WordNet is also considered as Ontology because it contains categorization of concepts into domains (e.g. food, architecture, sport, etc.).

2.2.2 Unstructured Resources

These include corpora which are collections of texts used for learning language models. These can be sense-annotated or raw unlabeled corpora.

2.2.2.1 Raw Corpora

Brown Corpus, British National Corpus (BNC), Wall Street Journal (WSJ) corpus, Gigaword corpus, etc are used as raw corpora for WSD.

2.2.2.2 Sense Annotated Corpora

SemCor [8] (the largest and most used sense tagged corpus), DSO corpus, Open Mind Word Expert data sets, SENSEVAL data sets, etc are used as sense annotated corpora for WSD.

2.2.3 Collocation Resources

These resources provide words that co occur with a tendency greater than chance (eg; bus, stop, station, network, communication). Examples of resources are: British National Corpus collocations, Collins Cobuild Corpus Concordance, Web1T corpus, etc.

2.3 WordNet

WordNet [7] is a machine readable dictionary(MRD), a sense inventory, widely used as knowledge source for WSD system, created and maintained at Princeton University (wordnet.princeton.edu). It encodes concepts in terms of sets of senses called synsets which express the similar meaning. WordNet 3.0 contains about 155,000 words organized in over 117,000 synsets. For eg; Synsets of noun ‘Car’ in WordNet 3.0 are:

1. [car^1_n , auto^1_n , automobile^1_n , machine^6_n , motorcar^1_n]

2. [car^2_n , railcar^1_n , railway car^1_n , railroad car^1_n]

3. [car^3_n , gondola^3_n]

4. [car^4_n , elevator car^1_n]

5. [cable car^1_n , car^5_n]

Superscript denotes the sense index and subscript the part of speech tag. Noun ‘Car’ has five senses, car^1_n is the first sense of noun ‘Car’ placed in its first synset. Each word sense identifies a single synset, and all word senses in a synset express the similar meaning. For each synset, WordNet provides the following information.

2.3.1 Gloss

A gloss is a textual definition of the synset with a set of usage examples. For e.g.; the gloss of first synset of noun ‘Car’ is defined in WordNet 3.0 as: a motor vehicle with four wheels; usually propelled by an internal combustion engine; "he needs a car to get to work".

2.3.2 Lexical and Semantic Relations

Lexical relations connect pair of word senses, where as semantic relations connect pair of synsets. Lexical relations are:

1. Antonymy: X is an antonym of Y if it expresses the opposite concept (e.g., $good^1_a$ is the antonym of bad^1_a).

2. Pertainymy: Adjective X pertains to noun (or, rarely, another adjective) Y (e.g., $dental^1_a$ pertains to $tooth^1_n$).

3. Nominalization: A noun X nominalizes a verb Y (e.g., $service^2_n$ nominalizes the verb $serve^4_v$).

Semantic relations are:
1. Hypernymy (also called kind-of or is-a): Y is a hypernym of X if every X is a (kind of) Y (motor $vehicle^1_n$ is a hypernym of car^1_n). Hypernymy holds between pairs of nominal or verbal synsets.

2. Hyponymy and troponymy: The inverse relations of hypernymy for nominal and verbal synsets, respectively.

3. Meronymy (also called part-of): Y is a meronym of X if Y is a part of X (e.g., flesh^3_n is a meronym of fruit^1_n). Meronymy holds for nominal synsets only.

4. Holonymy: Y is a holonym of X if X is a part of Y (the inverse of meronymy).

5. Entailment: A verb Y is entailed by a verb X if by doing X we must be doing Y (e.g., snore^1_v entails sleep^1_v).

6. Similarity: An adjective X is similar to an adjective Y (e.g., beautiful^1_a is similar to pretty^1_a).

7. Attribute: A noun X is an attribute for which an adjective Y expresses a value (e.g., hot^1_a is a value of temperature^1_n).

8. See also: A relation between adjectives (e.g., beautiful^1_a is related to attractive^1_a).

9. Domain: WordNet synsets have been semi automatically annotated with one or more domain labels such as food, architecture, sport, etc.

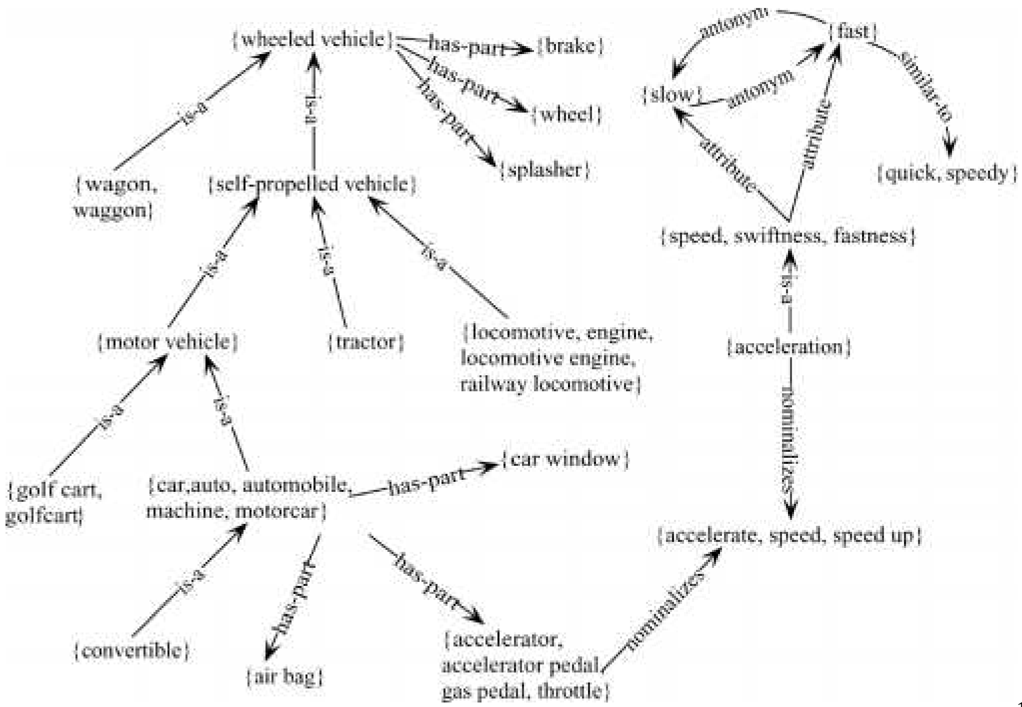


Fig. 2.1: An excerpt of the WordNet semantic network for a sense car^1_n

Because of its widespread diffusion within the research community, WordNet is considered as a standard sense inventory for English WSD. Following its success, WordNets for several languages have been developed and linked to the original Princeton WordNet.

2.4 Basic Approaches to WSD

The basic approaches to WSD are Dictionary-based and Corpus-based approaches.

2.4.1 Dictionary-based or Knowledge-based Approaches

These approaches rely on dictionaries, thesauri, ontologies, and other lexical knowledge bases. Semantic similarity measures between senses, overlap of definitions, etc are the methods used in this approach.

2.4.2 Corpus-based Approaches

These approaches can be supervised or unsupervised.

Supervised approaches use machine learning classifiers trained on examples which are manually annotated with the correct sense in accordance to predefined list of senses from a dictionary. These approaches are employed in Lexical sample task where only the restricted set of target words are required to be disambiguated. They have shown higher performance than unsupervised methods, but they require large amounts of training data to yield reliable results, and their coverage is limited to the words for which sense labeled data exist. Ng [29] estimates that a high accuracy domain independent supervised system for WSD would probably need a corpus of about 3.2 million sense tagged words. Creating sense tagged corpora manually is expensive which may not be feasible for every domain and languages. This problem suffer by supervised approaches is known as the knowledge-acquisition bottleneck.

Unsupervised approaches use raw unlabeled corpora including information contained in dictionary and lexical resources. They are employed in All-words WSD task, where all open-class words (nouns, verbs, adjectives, adverbs) are required to be disambiguated. They have potential to overcome the problems faced by supervised approaches. Unsupervised methods are much less costly to use and easier to transfer between domains. Fully unsupervised methods do not exploit any dictionary, and are based on the idea that the same sense of a word will have similar neighboring words. Fully unsupervised systems do the task of sense induction or sense discrimination, where word senses are induced from text by clustering word occurrences.

2.5 Graph Based Approaches to WSD

Graph based approaches to WSD is the emerging direction in WSD research. These approaches exploit graph structures to determine the most appropriate senses from words in context. These approaches represent the context as a graph where the nodes are word senses and the edges represent semantic connections between them.

Early graph based approaches are based on the notion of lexical chain. Approaches of Barzilay and Elhadad (1997) [9], Silber and McCoy (2003) [10], Galley and McKeown (2003) [11] are such approaches based on lexical chains. A lexical chain [12] [13] is a sequence of semantically related words $w_1,w_2,….,w_n$ in a text, such that w_i is related to w_{i+1} by a lexicosemantic relation (eg; is-a, has-part, etc). Lexical chains determine contexts and contribute to the continuity of meaning and the coherence of a discourse. Algorithms for computing lexical chains perform disambiguation before inferring which words are semantically related. Galley and Mckeown [11] performed a series of experiments on the set of 35,000 nouns from SemCor corpus with lexical chaining algorithms of Barzilay and Elhadad, Silber and McCoy, Galley and Mckeown, and reported WSD accuracy of 56.6%, 54.48%, 62.09% respectively on nouns.

Mihalcea [4] used a graph over word sequences, where the vertices are the different senses of each word, and are connected by edges, to senses of the k previous words in the sequence. Given a sequence of words $W=\{ w_1,w_2,….,w_n \}$, each word w_i with corresponding admissible labels $L_{wi} = \{L^1_{wi}, L^2_{wi}, ….., L^{Nwi}_{wi} \}$, a label $G = (V, E)$ is defined such that there is a vertex $v\in V$ for every possible label L^j_{wi}, $i=1…n$, $j=1… N_{wi}$. Label L_{wi} is the set of possible senses from the dictionary. Dependencies between pairs of labels are presented as directed or undirected edges $e\in E$, defined over the set of vertex pairs $V \times V$. Such dependencies can be learned from annotated data, or derived by other means. Author used WordNet gloss based similarity measure between two senses as label dependency. If similarity measure between two labels (senses) is zero, then there will be no edge between two labels. Otherwise, edge is drawn between labels and measure is assigned as a weight between two labels.

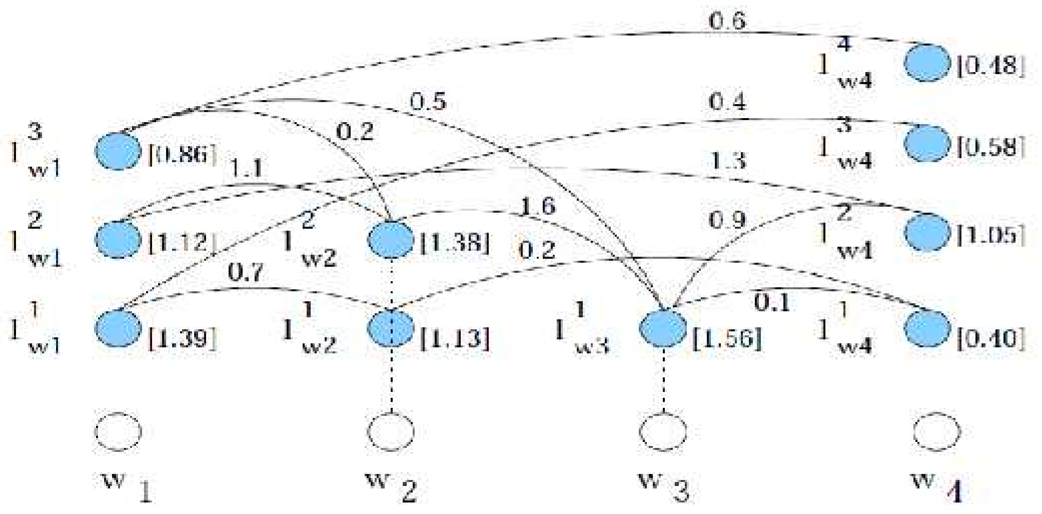


Fig. 2.2: Sample graph built on the set of possible labels for a sequence of four words

Author used PageRank to assign score to each label in the graph. The label for each word w_i whose score is highest among other, is assigned as the sense for disambiguation. Author obtained accuracy of 54.2% on the SENSEVAL-2 English all-words data set, and 64.2% on the same subset of SemCor nouns as used by Galley and Mckeown.

Navigli and Lapata [2] presented a generic graph-based algorithm which handles one sentence at a time and disambiguates words based entirely on the structure of the graph of WordNet. For a sentence to be disambiguated, they start with a graph G = (V, E) where V consists of each synset in WordNet corresponding to all the words in the sentence. They start with any vertex u in V and kept searching the WordNet graph upto the depth of certain length in depth-first search manner until they find another vertex v which was already present in V. Every time this happens, they add all intermediate nodes and edges on the path from u to v in G. For example, for the sentence "She drank some milk", omitting adverb "some", they obtained a following graph:

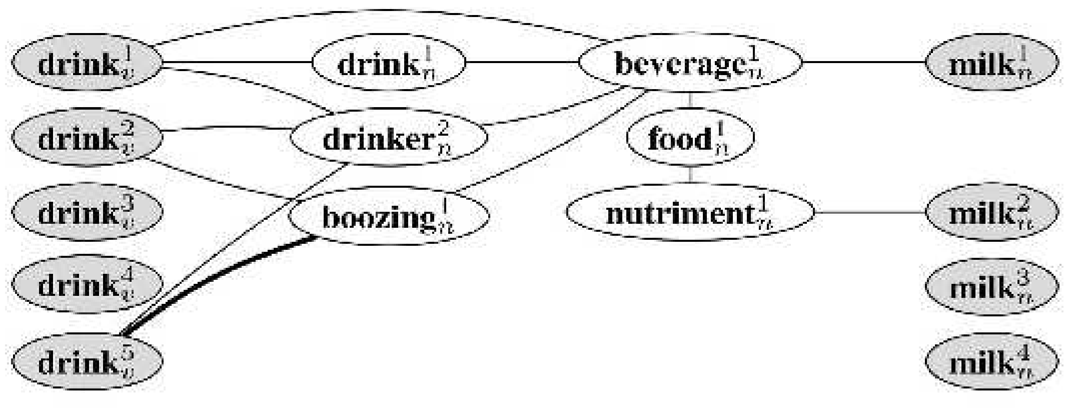


Fig. 2.3: Graph for the sentence “She drank some milk”

In order to find out the importance of vertex in the graph, they experimented using Degree, PageRank, HITS, KPP, and Betweeness centrality algorithms independently disambiguating text from SemCor. They obtained accuracy of 50.01% on all words disambiguation and concluded that Degree and PageRank are best measures.

Tsatsoronis, Varlamis, and Norvag [1] used WordNet for constructing semantic network by exploiting all available relations between word senses. Their method of constructing semantic network consists of two phases: initial phase, and expansion phase. In initial phase, words in a sentence and their senses are added to the network. In expansion phase, each node or sense in the network is visited using breadth-first search, and each sense S_i is expanded to S_j by adding edge (S_i, S_j) if there is a semantic relation between S_i and S_j in WordNet. This process is continued iteratively until there is a path between every pair of initial word nodes, and then network is considered connected. If network is not connected, then all words in the sentence cannot be disambiguated. During the process, if two words share the same senses, then a single sense node is added.

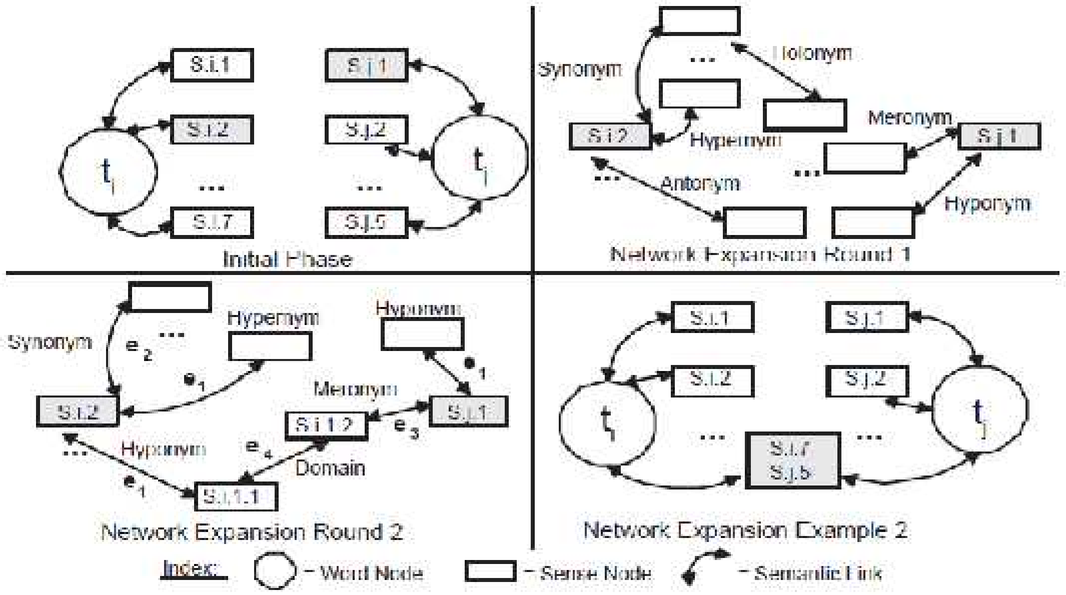


Fig. 2.4: Semantic Network construction in the approach of Tsatsoronis, Varlamis, and Norvag

Weight is assigned to each edge which represents degree of relatedness between two senses. For every edge e_{kj} in the graph, they assigned weight with the quantity ETF(e_{kj}).INF(e_{kj}), where ETF(e_{kj}) is Edge Type Frequency, and INF(e_{kj}) is Inverse Node Frequency for edge e_{kj} respectively. ETF is similar to TF (Term Frequency) in IR, and INF is similar to IDF (Inverse Document Frequency) in IR. ETF for edge e_{kj} is the frequency of outgoing edges from node k that are of the same type as $e_{kj.}$ INF for edge e_{kj} is the log inverse of the frequency of nodes that have outgoing edges of the same type as $e_{kj.}$ They evaluated their method on English all-words data set from Senseval-2 and Senseval-3, and obtained accuracy of 58.8% and 56.7% respectively with PageRank.

CHAPTER 3

IMPLEMENTATION

3.1 Overview of the phases in Implemented WSD System

The implemented WSD system consists of phases shown in figure below.

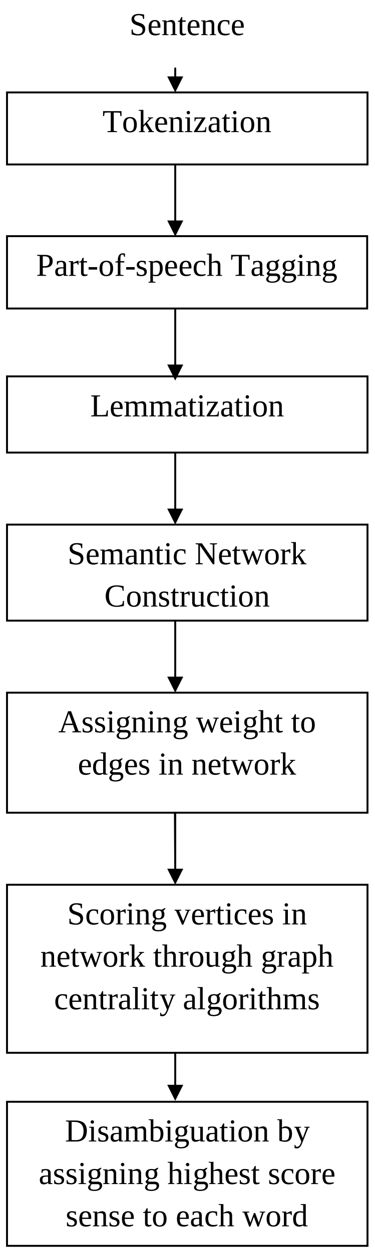

Sentence
Tokenization
Part-of-speech Tagging
Lemmatization
Semantic Network Construction
Assigning weight to edges in network
Scoring vertices in network through graph centrality algorithms
Disambiguation by assigning highest score sense to each word

Fig 3.1: Phases in implemented WSD system

In first stage, sentence is tokenized into words. This is done by using OpenNLP which is NLP toolkit written in Java available freely from opennlp.sourceforge.net. OpenNLP is used for doing various NLP tasks such as sentence segmentation, tokenization, part-of-speech tagging, chunking, parsing, etc. Thus, words are tagged with using part-of-speech tagger from OpenNLP. Only nouns, verbs, adjectives, and adverbs are selected. Then these words are reduced to their base form, process known as lemmatization (eg; 'was' is reduced to 'be', 'apples' is reduced to 'apple'). This is be done by using WordNet. A Java based library called JWNL (Java WordNet Library) which is available freely is used for accessing the WordNet dictionary. The base form of a word is called a lemma. The lemma of a word is used for accessing senses of a word from WordNet. JWNL contains programming classes and functions which are used for doing lemmatization of a word, accessing senses of a word, fetching semantically related senses for a given sense, etc. Thus a semantic network containing senses of selected words is constructed using JWNL and weight is assigned to each edge by ETF-INF. Using graph centrality algorithm, score is assigned to each node in the network. Finally, for each word, their highest score sense is assigned and sense disambiguation is done.

3.2 Semantic Network Construction

As in the method of Tsatsoronis, Varlamis, and Norvag [1], the method has two phases for constructing semantic network of senses: initial phase and expansion phase.

Consider the sequence of words $W=\{w_1,w_2,\ldots,w_n\}$ in a sentence to be disambiguated. The initial phase is initiated by number of graphs, each containing only one sense for a word w_i For example, consider $\{w_1, w_2\}$ is the sequence of two words in a sentence to be disambiguated, and if w_1 has 3 senses and w_2 has 2 senses. Then initial phase will contain 5 graphs each containing a single but different sense from w_1 and w_2.

In expansion phase, each sense (unexpanded sense) in graphs is expanded by adding semantically related senses from WordNet. The sense S_i is expanded to S_j (or more senses) by adding edge (S_i, S_j) in graph, if there is a semantic relation between S_i and S_j in WordNet. Before adding the new sense S_j , it is searched over all graphs. If S_j is found in different graph other than S_i, then those two graphs are merged at common vertex S_j

(also, if any two senses belong to same synset, they are placed in same vertex). This process of expanding vertices in each graph is continued until a connected graph G containing senses of all words in W is obtained.

If such a graph containing senses of all words in W is not obtainable, then graphs are allowed to grow to maximum fixed size, and one containing senses from most words in W is considered appropriate for applying node ranking algorithms. The remaining words whose senses are not in the semantic graph are assigned their first sense, if WordNet is used as a reference dictionary. Because in WordNet, first sense for each word is the most frequently used sense (MFS) in Brown corpus, and many researchers have used MFS to break a tie between two senses, in case two senses have same score [11].

Thus, construction of semantic network is initiated from number of single vertex sense graphs, and these graphs are continuously expanded and merged in order to make a connected network that will contain senses of all words to be disambiguated. The network construction process for two words w_1, w_2 in a sentence is described by figure below.

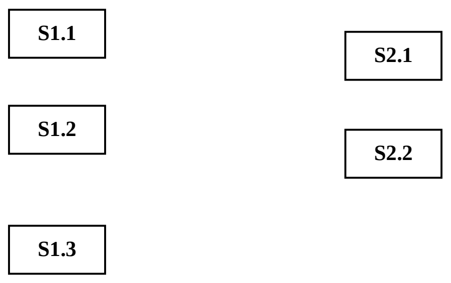


Fig. 3.2 : Initial phase containing 5 single vertex graphs, which are senses of words w_1, w_2. S1 are senses of w_1, and S2 are senses of w_2.

In the above figure, S1.1, S1.2, S1.3 are three different senses of w_1 and S2.1, S2.2 are two different senses of w_2 . In initial phase, each different sense of words are considered vertices of different graphs. Thus, there are five graphs each containing only one node in the initial phase of network construction.

The expansion phase is carried on by number of iterations. In each iteration, vertices in each graph are visited. If a vertex has a degree of zero or one then semantically associated senses of that sense vertex is fetched from WordNet. A vertex having degree of one or zero is expanded by adding edges with its associated senses. For example, first sense of a word car represented as car^1_n which is defined as "a motor vehicle with four wheels; usually propelled by an internal combustion engine" in WordNet has following semantically associated senses: motor vehicle^1_n, convertible^1_n, air bag^1_n, accelerator^1_n, car window^1_n, wing$^{10}_n$. If a vertex car^1_n has to be expanded then 6 edges with vertices: motor vehicle^1_n, convertible^1_n, air bag^1_n, accelerator^1_n, car window^1_n, wing$^{10}_n$ are added to vertex car^1_n. The process is described in figures below, where S1.1.1 and S1.1.2 are senses expanded to S1.1. If newly added sense in the graph is found in any other graph or if synset of newly added sense is found in any other graph then those two graphs are merged at common synset or sense. This process is continued until a graph containing senses of all words being disambiguated is obtained.

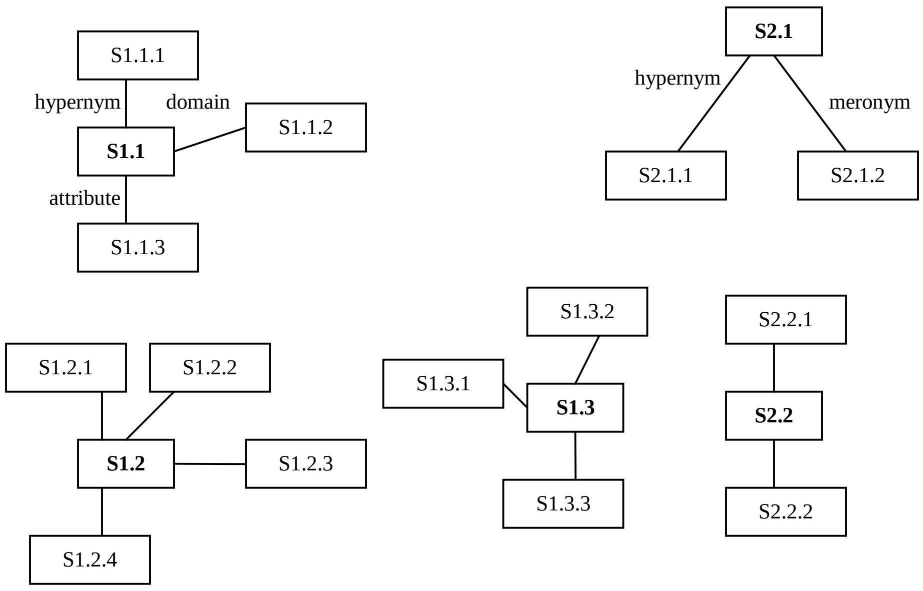
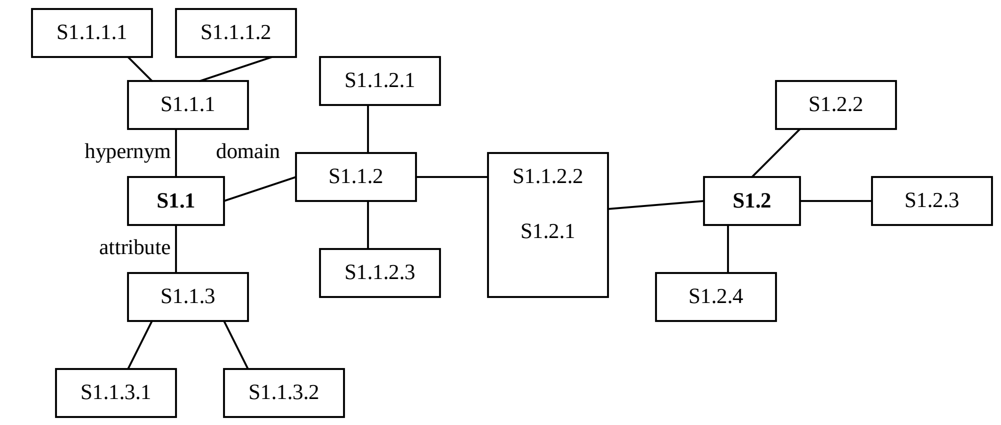


Fig. 3.3 : Expansion phase Round 1, single vertex graphs are expanded



Two graphs with initial vertex S1.1 and S1.2 will be merged if S1.1.2.2 and S1.2.1 are same senses

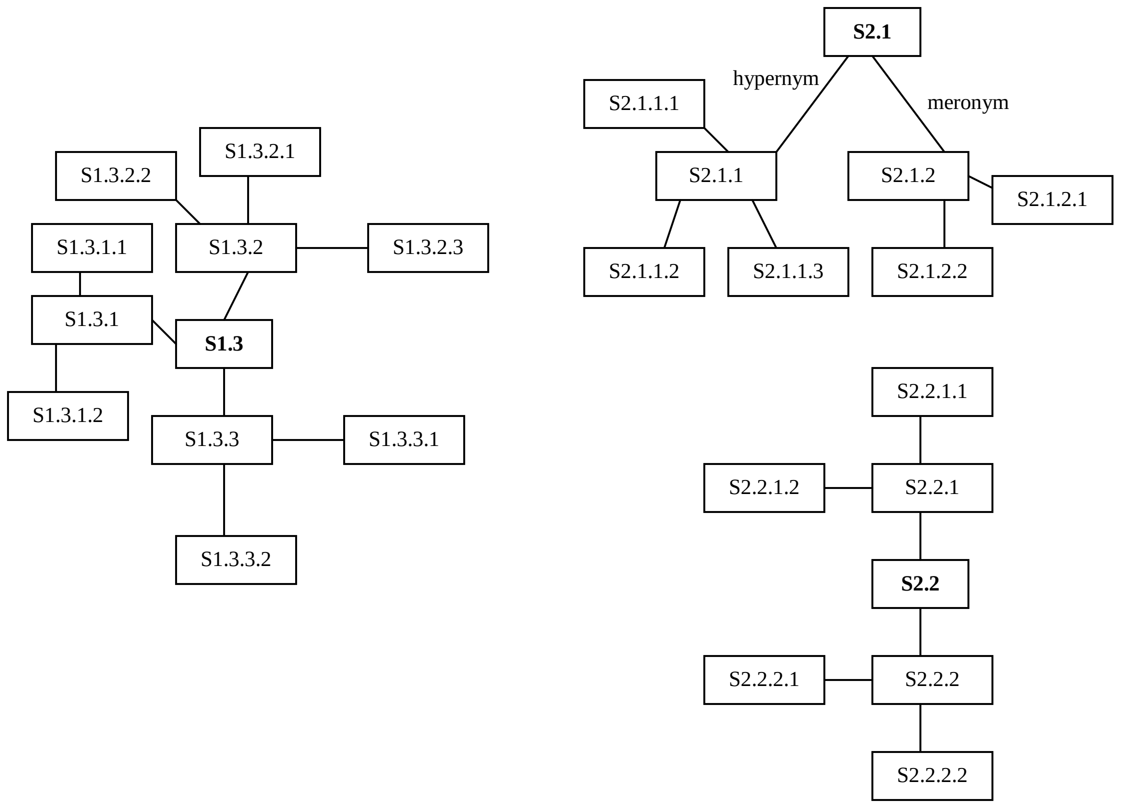


Fig. 3.4: Expansion phase Round 2, graphs are on continuous expansion

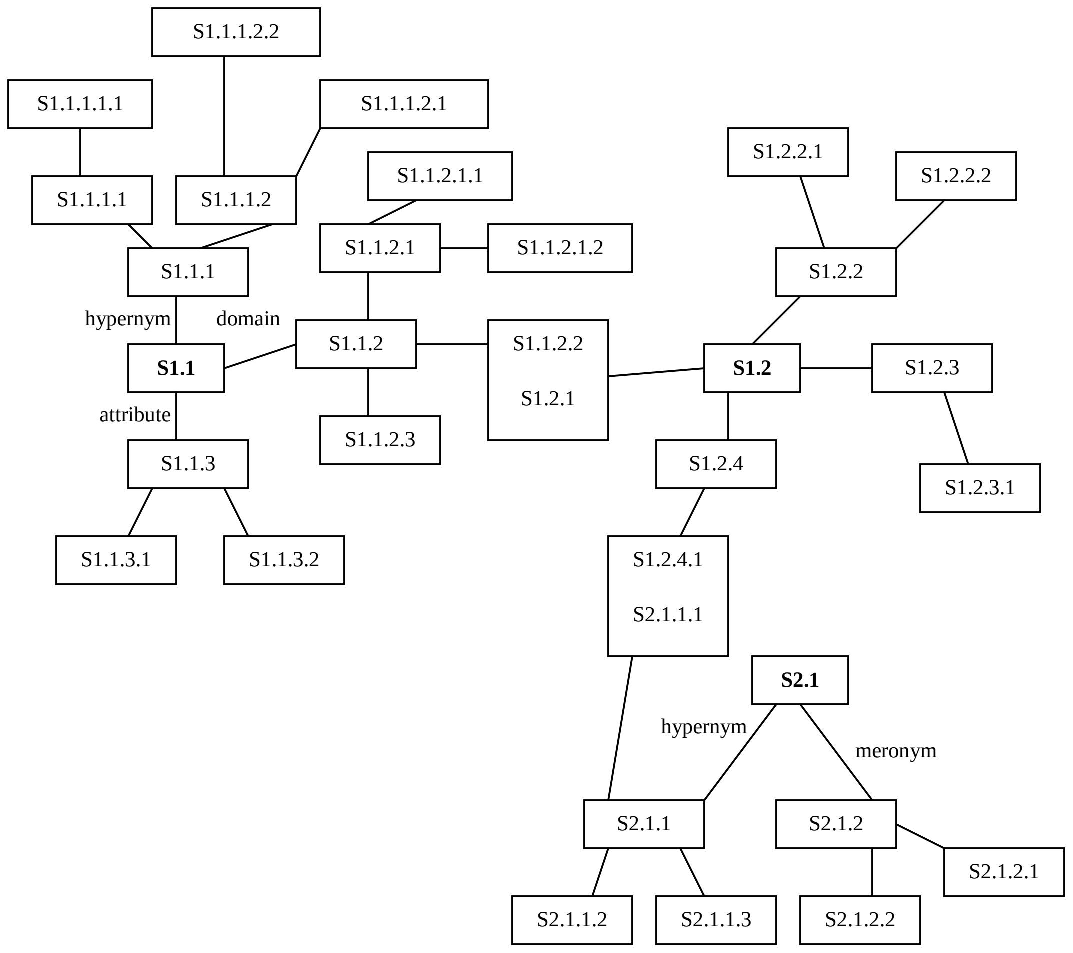


Fig. 3.5: Expansion phase Round 3, a graph with vertices S1.1, S1.2 and a graph with vertex S2.1 will be merged if S1.2.4.1 and S2.1.1.1 are same senses

Graph obtained in fig. 3.5 consists of senses (S1.1, S1.2, S2.1) of all words (w_1, w_2) in a sentence which is to be disambiguated. So, expansion phase is stopped, if such a graph containing senses of all words is obtained. Hence, there is no need to expand other graphs. In practice, a graph with more than 1000 vertices will be obtained. In order to determine which sense to assign to each word, for example sense S1.1 or sense S1.2 to word w_1, graph centrality algorithm is applied. If a score given by centrality algorithm for vertex S1.2 is higher than S1.1 then sense S1.2 is assigned to w_1. Thus, network construction process can be summarized in following steps.

1. In initial phase, graphs with single vertex are constructed, where vertex in each graph is a sense of the word being disambiguated.

2. In expansion phase, each vertex in graphs is visited. If a vertex has a degree of zero or one, it is expanded by adding semantic edge with a sense from WordNet. If newly added sense is found in any other graph, then graphs will be merged.

3. Step 2 is repeated until a graph containing senses of all words being disambiguated in a sentence is obtained. If such a graph is not obtained, then step 2 is continued until all graphs attain certain fixed size. Finally, a graph having senses of most words is chosen for applying graph centrality algorithm.

3.3 Assigning Edge Weights in Network with ETF-INF

ETF-INF is the concept borrowed from TF-IDF in IR, assuming type of edge in the semantic network similar to token in a document, and node in the network similar to document. The weightage scheme was first introduced by Tsatsoronis, Vazirgiannis, Androutsopoulos [3] for assigning weights in large semantic network of senses for WSD.

ETF (Edge Type Frequency) for edge e_{kj} is the frequency of outgoing edges from node k that are of the same type as e_{kj}, which is obtained as:

ETF(e_{kj})= (number of outgoing edges from node k of type e_{kj})/(number of outgoing edges from node k)

$$= \frac{|\{e_{ki}:type(e_{ki})=type(e_{kj})\}|}{|\{e_{ki}\}|}$$

Consider a sub graph in the network:

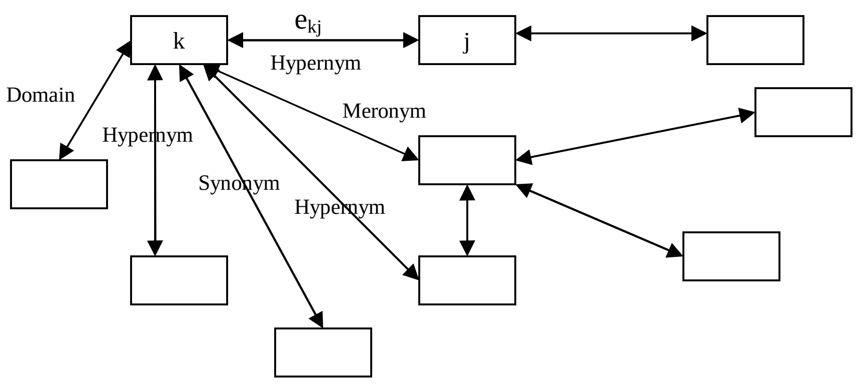


Fig. 3.6: A sub graph in the network

In the above figure,

ETF(e_{kj}) = (number of outgoing edges from node k of type hypernym) / (number of outgoing edges from node k)

=3/6

=0.5

INF (Inverse Node Frequency) for edge e_{kj} is the log inverse of the frequency of nodes that have outgoing edges of the same type as e_{kj}, which is obtained as:

$$\mathrm{INF}(e_{kj}) = \log \frac{N+1}{N_{type(ekj)}}$$

where, N is the total number of nodes in the network and $N_{type(ekj)}$ is the number of nodes that have outgoing edges of the same type as e_{kj}.

ETF-INF for edge e_{kj} from node k to j will be different from ETF-INF for edge e_{jk} from node j to k.

3.4 Graph Based Centrality Algorithms

The following graph centrality algorithms are implemented and experimented on the semantic network: Indegree, PageRank, and HITS.

3.4.1 Indegree Centrality

It measures vertex importance by its degree. In unweighted graph, indegree of a vertex v is the number of edges terminating in a vertex v. In weighted graph, the indegree of a vertex is the sum of the weight of edges incident on that vertex.

$$\text{Indegree}(V_i) = \sum_{V_j \in In(V_i)} W_{ij}$$

The complexity of obtaining indegree for each vertex in graph is O(n).

3.4.2 PageRank

PageRank [15] determines the relevance of a node v recursively based on a Markov chain model. For unweighted graph, the PageRank score for a vertex V_a is a recursive function:

$$P(V_a) = (1-d) + d \sum_{V_b \in In(V_a)} \frac{P(V_b)}{|out(V_b)|}$$

where, $In(V_a)$ is the set of vertices (predecessors) that point to V_a, $Out(V_b)$ is the set of vertices (successors) that V_b points to, and d is a parameter set to 0.85 [15].

For weighted graph, the PageRank score [4] for vertex V_a is given by:

$$WP(V_a) = (1-d) + d \sum_{V_b \in In(V_a)} \frac{W_{ba} \; WP(V_b)}{\sum_{V_c \in Out(V_b)} W_{bc}}$$

where, W_{ba} is the weight of the edge connecting vertices V_b and V_a . Starting from arbitrary values assigned to each vertex, the computation is iterated until convergence below a given threshold is achieved. After running the algorithm, a score will be associated with each vertex.

Algorithm

```
1: $WP_i^{(0)}=1$, for all vertices i=1,2,…n
2: for k=1,2,.. do
3:     for i=1,2, ….n do
4:         $WP_i^{(k)} = (1-d) + d \sum_{j \in In(i)} \frac{W_{ji}}{\sum_{l \in Out(j)} W_{jl}} WP_j^{(k-1)}$
5:     end for
6:     if $\| WP^{(k)} - WP^{(k-1)} \| <$ tolerance (for all i) then
7:         break
8:     end if
9: end for
```

The complexity of PageRank algorithm is $O(n^2)$.

3.4.3 HITS

HITS stands for Hypertext Induced Topic Search, introduced by Kleinberg [16]. It is based on identifying the authorities (the most important pointed nodes in a graph) and the hubs (the nodes that point to authorities). Each graph node has a pair of values: hub and authority score. In unweighted graph, these values for a node p, can be obtained as:

$$\text{authority}(p) = \sum_{q \in In(p)} \text{hub}(q)$$

$$\text{hub}(p) = \sum_{r \in Out(p)} \text{authority}(r)$$

where, In(p) is the set of nodes that point to p, and out(p) is the set of nodes that p points to. In weighted graph, these values are obtained as:

$$\text{authority}(p) = \sum_{q \in In(p)} W_{q,p} \, \text{hub}(q)$$

$$\text{hub}(p) = \sum_{r \in Out(p)} W_{p,r} \, \text{authority}(r)$$

where, $W_{q,p}$ is the weight of the edge from node q to p. Initially, these values are set to 1. Then algorithm is run iteratively to update the hub and authority score. After a number of iterations, the authority and the hub values for each node converge when normalization is used, dividing at each iteration, each authority value by the sum of the authority values and each hub value by the sum of hub values. Finally, the node with the highest authority value is selected as the most appropriate sense for each word.

Algorithm

```
1: $auth_i^{(0)} = 1$, $hub_i^{(0)} = 1$, for all vertices i=1,2,…..n
2: for k=1,2,..  do
3:    for i=1,2, ….n      do
4:         $auth_i^{(k)} = \sum_{j \in In(i)} W_{j,i} \; hub_j^{(k-1)}$
5:         $hub_i^{(k)} = \sum_{j \in Out(i)} W_{i,j} \; auth_j^{(k-1)}$
6:         $authSum_i^{(k)} = authSum_i^{(k)} + auth_i^{(k)}$
7:         $hubSum_i^{(k)} = hubSum_i^{(k)} + hub_i^{(k)}$
8:    end for
9:    for i=1,2, ….n      do
10:        $auth_i^{(k)} = auth_i^{(k)} / authSum_i^{(k)}$
11:        $hub_i^{(k)} = hub_i^{(k)} / hubSum_i^{(k)}$
12:   end for
13:   if   $\| auth^{(k)} - auth^{(k-1)} \| <$ tolerance and $\| hub^{(k)} - hub^{(k-1)} \| <$ tolerance (for all i) then
14:       break
15:    end if
16: end for
```

The complexity of HITS algorithm is $O(n^2)$.

CHAPTER 4

EXPERIMENTS AND EVALUATIONS

According to Ide and Veronis [14], two possible types of evaluations for WSD system are in vitro and in vivo evaluation. The evaluation of WSD systems as a stand-alone and independent application is in vitro evaluation, whereas the evaluation of WSD as a module embedded in applications (such as IR, MT) is in vivo evaluation. In this dissertation, system is evaluated considering system as a stand-alone and independent application, that is, in vitro evaluation.

In vitro evaluation is done by disambiguating words from manually sense tagged corpora, and calculating Precision, Recall and F_1-measure in order to measure system performance.

$$\text{Precision (P)} = \frac{\text{Number of words correctly disambiguated by the system}}{\text{Number of words disambiguated by the system}}$$

$$\text{Recall (R)} = \frac{\text{Number of words correctly disambiguated by the system}}{\text{Number of words to be disambiguated by the system}}$$

F_1-measure called balanced F-score weights precision and recall equally, given by:

$$F_1 = \frac{2\,P\,R}{P + R}$$

4.1 Data

Subset of SemCor [8] and corpora from Senseval-2 and Senseval-3 are used for the evaluation of the system.

SemCor consists of about 221,000 sense tagged words and 186 files with each corresponding to a file in Brown corpus. This dissertation uses WordNet version 3.0 for implementing the system, so SemCor tagged with WordNet 3.0 senses is downloaded from www.cse.unt.edu/~rada/downloads.html . The 10 files containing 9529 open class words in 1079 sentences are used for evaluation purpose.

Senseval (www.senseval.org) is an international word sense disambiguation competition, held every three years since 1998. Its objective is to perform a comparative evaluation of WSD systems in several kinds of tasks. Senseval-2 and Senseval-3 data sets for English all words disambiguation, tagged with WordNet 3.0 senses are downloaded from www.cse.unt.edu/~rada/downloads.html . Table below shows the statistics of data sets for evaluation purpose.

Data set	Nouns	Verbs	Adjectives	Adverbs	Total
SemCor	4569	2509	1460	991	9529
SENSEVAL-2	1015	525	390	259	2189
SENSEVAL-3	847	679	308	13	1847

Table 4.1: Statistics of test data

4.2 The optimal size for graphs to expand

During the network construction process (initial and expansion phases), graphs are continuously expanded and merged in order to make a connected network that will contain senses of all words to be disambiguated. But after number of expansions, if such a network is not obtainable then expansion process must be discontinued. Because on continuous expansion graphs may contain over 100,000 senses which do not help to increase disambiguation accuracy. Thus, in order to obtain optimal size for graphs to expand, 1000 sentences from SemCor corpus is randomly taken and a semantic network is constructed for each sentence. It is found that size at which network contains senses of all words in a sentence do not exceed above 5000 vertices. Hence, the size of 5000 vertices is considered optimal for graphs to expand during network construction process.

4.3 Evaluation of the system

For each sentence from test data set including 10 files from SemCor, and data sets from Senseval-2 and Senseval-3, semantic network with unweighted edges and with weighted edges assigned by ETF-INF are constructed. Indegree, PageRank and HITS are applied to score vertices, and sense disambiguation is done for each word in the sentence by assigning highest score sense for each word. The implemented system can disambiguate all open class words in test data sets, hence Precision, Recall, and F_1-measure are equal. These measures calculated in percentage for different centrality algorithms in the system are summarized below.

4.3.1 Results obtained with unweighted semantic network

	Noun	Verb	Adjective	adverb	All (P/R/F_1)
Senseval-2					
Indegree	67.78	35.43	61.03	67.18	58.75
PageRank	67.19	35.43	56.41	69.11	57.88
HITS	64.63	32	54.61	69.11	55.55
Senseval-3					
Indegree	56.2	40.35	59.42	100	51.22
PageRank	55.6	39.76	58.12	100	50.51
HITS	54.31	28.13	58.12	100	45.64
SemCor					
Indegree	61.39	41.53	66.37	72.86	58.12
PageRank	61.24	41.17	66.71	72.96	58.01
HITS	59.62	34.32	65.21	73.06	55.21

Table 4.2: Results obtained with unweighted semantic network on test data sets

4.3.2 Results obtained with weighted semantic network

	Noun	Verb	Adjective	Adverb	All (P/R/F_1)
Senseval-2					
Indegree	62.56	36	60.51	66.80	56.33
PageRank	65.91	35.05	61.28	66.02	57.7
HITS	63.74	33.52	59.23	66.02	55.96
Senseval-3					
Indegree	54.1	37.56	60.71	100	49.43
PageRank	54.9	38.73	55.84	100	49.43
HITS	54.55	32.55	54.87	100	46.83
SemCor					
Indegree	60.67	41.73	65.75	72.55	57.7
PageRank	60.28	40.77	63.29	71.54	56.77
HITS	58.66	35.79	61.78	71.44	54.44

Table 4.3: Results obtained with weighted semantic network on test data sets

4.4 Discussion

Experiments reveal that Indegree and PageRank centrality have better disambiguation accuracy than HITS. Disambiguation accuracy of 58.12% in SemCor, 58.75% in Senseval-2, and 51.22% in Senseval-3 are obtained by Indegree centrality in network with unweighted edges.

Further, in order to determine how diversely three methods assign senses, their pair-wise inter-agreement (using unweighted edge network) in three test data sets are determined. Pair-wise inter-agreement is the percentage of the same sense assignment to the total sense assignments performed.

Pairwise inter-agreement	SemCor	Senseval-2	Senseval-3
Indegree-PageRank	88.65	87.17	90.02
Indegree-HITS	75.16	75.07	71.61
PageRank-HITS	79.31	80.37	75.60

Table 4.4: Pair-wise inter-agreement in test data sets

Above statistics reveal that Indegree and PageRank have highest inter-agreement of senses assignment, which indicates Indegree and PageRank are closely related to each other. This is reasonable because PageRank is considered as an extended version of Degree centrality.

4.5 Limitation of the proposed method

The method proposed in the dissertation to construct a semantic network of sense requires a machine readable dictionary (MRD) as a knowledge source which provides explicit semantic relations between senses of words. Method does a disambiguation through a connected graph containing senses of all words in a sentence. If such a graph is not obtainable, then graphs are allowed to expand to a certain size, and graph containing senses of most words is chosen for applying graph centrality algorithm. The remaining words whose senses are not in the semantic graph are assigned their first sense, which is the most frequent sense (MFS) in the Brown Corpus.

CHAPTER 5

CONCLUSION AND FUTURE STUDY

5.1 Conclusion

Graph based unsupervised approach to WSD is a promising direction in WSD research as it has potential to overcome the knowledge acquisition bottleneck, the problem faced by supervised systems. Because graph based unsupervised approaches do not require manually sense tagged data for machine learning as in supervised approaches. If a lexical knowledge source like WordNet keeps on developing, enriching with many semantic relations between senses, these graph based approaches using such a knowledge source are expected to do disambiguation with higher accuracies.

The approach with simple degree centrality is faster and accurate than other centrality measures. Hence, the approach is suitable for large scale and domain independent disambiguation. The edges in semantic network do not necessary required to be weighted. Because, semantic network is a connected graph containing all related senses from knowledge source that can help disambiguating the each word in sentence. Only, in disconnected sense graph containing relatively few vertices, weight assigned between senses is significant.

5.2 Future study

The method discussed in this dissertation use semantic network generated from WordNet for WSD. The accuracy of disambiguation will certainly be increased, if WordNet could further be enriched by adding more semantic relations between senses, more organization of concepts, etc. For example, Navigli and Lapata [2] used extended version of WordNet [24] which was extended by adding 60,000 collocational relations compiled from Oxford collocations, Longman Language Activator dictionaries, and collocation web sites, and shown increased in accuracy of disambiguation. Similarly, Navigli and Ponzetto [25] extended WordNet by adding large amounts of semantic relations (between nouns only) from Wikepedia and shown that system could perform competitively with supervised approaches in coarse grained all words WSD. The future study can be finding out more explicit relation between senses, so that more denser network of senses can be constructed, and better disambiguation accuracy can be obtained.

References

[1] Tsatsaronis, G., Varlamis, I., Norvag, K.: *An Experimental Study on Unsupervised Graph-based Word Sense Disambiguation.* In: Proc. of 11th International Conference on Computational Linguistics and Intelligent Text Processing, CICLing, 184–198 (2010)

[2] Navigli, R., Lapata, M.: *An experimental study on graph connectivity for unsupervised Word Sense Disambiguation.* IEEE Transactions on Pattern Analysis and Machine Intelligence, 32(4):678-692 (2010)

[3] Tsatsaronis, G., Vazirgiannis, M., Androutsopoulos, I.: *Word sense disambiguation with spreading activation networks generated from thesauri.* In: Proc. of IJCAI, pp. 1725–1730 (2007)

[4] Mihalcea, R.: *Unsupervised large-vocabulary word sense disambiguation with graph based algorithms for sequence data labeling.* In: HLT (2005)

[5] Navigli, R.: *Word sense disambiguation: A survey.* ACM Computing Surveys 41(2), Article 10 (2009)

[6] Agiree, E., Edmonds, P., Eds. 2007. *Word Sense Disambiguation: Algorithms and Applications*. Springer, New York, NY

[7] Fellbaum, C., Ed. 1998. *WordNet: An Electronic Database*. MIT Press, Cambridge, MA.

[8] G. Miller, C. Leacock, R. Bunker. 1993. *A semantic concordance.* In Proceedings of the 3rd DARPA Workshop on Human Language Technology, Plainsboro, New Jersey.

[9] Barzilay, R. And Elhadad, M. 1997. *Using lexical chains for text summarization.* In proceedings of the ACL Workshop on Intelligent Scalable Text Summarization (Madrid,Spain). 10–17.

[10] Silber, H. G. AND Mccoy, K. F. 2003. *Efficient text summarization using lexical chains.* In Proceedings of the 5th International Conference on Intelligent User Interfaces (New Orleans, LA). 252–255.

[11] Galley, M. AND Mckeown, K. 2003. *Improving word sense disambiguation in lexical chaining*. In Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI, Acapulco, Mexico).1486–1488.

[12] Morris, J. And Hirst, G. 1991. *Lexical cohesion computed by thesaural relations as an indicator of the structure of text.* Computat. Ling. 17, 1.

[13] Hirst, G. And ST-Onge, D. 1998. *Lexical chains as representations of context for the detection and correction of malapropisms.* In *WordNet: An Electronic Lexical Database*, C. Fellbaum, Ed. MIT Press, Cambridge,MA, 305–332.

[14] Ide, N. and Veronis, J., *Word sense disambiguation: the state of the art.* Computational Linguistics, 24(1):1–40, 1998.

[15] Brin, S. and Page, L., *Anatomy of a Large-Scale Hypertextual Web Search Engine*, Proc. Seventh Conf. World Wide Web, 107-117, 1998.

[16] Kleinberg, J.M., *Authoritative Sources in a Hyperlinked Environment*, Proc. Ninth Symp. Discrete Algorithms, 668-677, 1998.

[17] Carpuat, M. and Wu, D. 2007. *Improving statistical machine translation using word sense disambiguation.* In Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL, Prague, Czech Republic). 61–72.

[18] Chan, Y. S., Ng, H. T., and Chiang, D. 2007. *Word sense disambiguation improves statistical machine translation.* In Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics(Prague, Czech Republic). 33–40.

[19] SANDERSON, M. 1994. *Word sense disambiguation and information retrieval.* In Proceedings of the Special Interest Group on Information Retrieval (SIGIR, Dublin, Ireland). 42–151.

[20] Clough, P. & Stevenson, M. 2004. *Cross-language information retrieval using EuroWordNet and word sense disambiguation.* Advances in Information Retrieval, 26th European Conference on IR Research (ECIR 2004), Sunderland, UK, 327–337.

[21] Vossen, Piek, German Rigau, Inaki Alegria, Eneko Agirre, David Farwell & Manuel Fuentes. 2006. *Meaningful results for information retrieval in the MEANING project.* Proceedings of the Third Global WordNet Conference, Jeju Island, Korea.

[22] Berendt, B. and Navigli, R. 2006. *Finding your way through blogspace: Using semantics for cross-domain blog analysis.* In Proceedings of the AAAI Spring Symposium 2006 (AAAIS) on Computational Approaches to Analysing Weblogs (CAAW, Palo Alto, CA).1–8.

[23] Hovy, Eduard, Ulf Hermjakob & Deepak Ravichandran. 2002. *A question/answer typology with surface text patterns.* Proceedings of the DARPA Human Language Technology Conference (HLT), San Diego, CA.

[24] Navigli, R., *Semi-Automatic Extension of Large-Scale Linguistic Knowledge Bases*, Proc. 18th Florida Artificial Intelligence Research Soc. Conf., pp. 548-553, 2005.

[25] Navigli, R., and Ponzetto, S., *Knowledge-rich Word Sense Disambiguation Rivaling Supervised Systems*. In Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pp. 1522-1531, 2010

[26] Weaver, W. *Machine translation of languages*, John Wiley and Sons, 1949.

[27] Yorick Wilks, *Formal semantics of natural language*, Cambridge University Press, Cambridge, UK, 1975.

[28] LESK, M. 1986. *Automatic sense disambiguation using machine readable dictionaries: How to tell a pine cone from an ice cream cone.* In Proceedings of the 5th SIGDOC (New York, NY). 24–26.

[29] NG, T. H. 1997. *Getting serious about word sense disambiguation.* In Proceedings of the ACL SIGLEX Workshop on Tagging Text with Lexical Semantics: Why, What, and How? (Washington D.C.). 1–7.

Appendix A

Sample Texts from Test Corpus

```
<contextfile concordance=brown>
<context filename=br-a01 paras=yes>
<p pnum=1>
<s snum=1>
<wf cmd=ignore pos=DT>The</wf>
<wf cmd=done rdf=group pos=NNP lemma=group wnsn=1 lexsn=1:03:00::
pn=group>Fulton_County_Grand_Jury</wf>
<wf cmd=done pos=VB lemma=say wnsn=1 lexsn=2:32:00::>said</wf>
<wf cmd=done pos=NN lemma=friday wnsn=1 lexsn=1:28:00::>Friday</wf>
<wf cmd=ignore pos=DT>an</wf>
<wf cmd=done pos=NN lemma=investigation wnsn=1
lexsn=1:09:00::>investigation</wf>
<wf cmd=ignore pos=IN>of</wf>
<wf cmd=done pos=NN lemma=atlanta wnsn=1 lexsn=1:15:00::>Atlanta</wf>
<wf cmd=ignore pos=POS>'s</wf>
<wf cmd=done pos=JJ lemma=recent wnsn=2 lexsn=5:00:00:past:00>recent</wf>
<wf cmd=done pos=NN lemma=primary_election wnsn=1
lexsn=1:04:00::>primary_election</wf>
<wf cmd=done pos=VB lemma=produce wnsn=4 lexsn=2:39:01::>produced</wf>
<punc>``</punc>
<wf cmd=ignore pos=DT>no</wf>
<wf cmd=done pos=NN lemma=evidence wnsn=1 lexsn=1:09:00::>evidence</wf>
<punc>"</punc>
<wf cmd=ignore pos=IN>that</wf>
<wf cmd=ignore pos=DT>any</wf>
<wf cmd=done pos=NN lemma=irregularity wnsn=1 lexsn=1:04:00::>irregularities</wf>
<wf cmd=done pos=VB lemma=take_place wnsn=1 lexsn=2:30:00::>took_place</wf>
<punc>.</punc>
</s>
</p>
<p pnum=2>
<s snum=2>
<wf cmd=ignore pos=DT>The</wf>
<wf cmd=done pos=NN lemma=jury wnsn=1 lexsn=1:14:00::>jury</wf>
<wf cmd=done pos=RB lemma=far wnsn=2 lexsn=4:02:00::>further</wf>
<wf cmd=done pos=VB lemma=say wnsn=1 lexsn=2:32:00::>said</wf>
<wf cmd=ignore pos=IN>in</wf>
<wf cmd=done pos=NN lemma=term wnsn=2 lexsn=1:28:00::>term</wf>
<wf cmd=done pos=NN lemma=end wnsn=2 lexsn=1:28:00::>end</wf>
<wf cmd=done pos=NN lemma=presentment wnsn=1
lexsn=1:04:00::>presentments</wf>
```

<wf cmd=ignore pos=IN>that</wf>
<wf cmd=ignore pos=DT>the</wf>
<wf cmd=done rdf=group pos=NNP lemma=group wnsn=1 lexsn=1:03:00:: pn=group>City_Executive_Committee</wf>
<punc>,</punc>
<wf cmd=ignore pos=WDT>which</wf>
<wf cmd=done pos=VB lemma=have wnsn=4 lexsn=2:40:04::>had</wf>
<wf cmd=done pos=JJ lemma=overall wnsn=2 lexsn=5:00:00:gross:00>over-all</wf>
<wf cmd=done pos=NN lemma=charge wnsn=6 lexsn=1:04:03::>charge</wf>
<wf cmd=ignore pos=IN>of</wf>
<wf cmd=ignore pos=DT>the</wf>
<wf cmd=done pos=NN lemma=election wnsn=1 lexsn=1:04:01::>election</wf>
<punc>,</punc>
<punc>``</punc>
<wf cmd=done pos=VB lemma=deserve wnsn=1 lexsn=2:42:00::>deserves</wf>
<wf cmd=ignore pos=DT>the</wf>
<wf cmd=done pos=NN lemma=praise wnsn=1 lexsn=1:10:00::>praise</wf>
<wf cmd=ignore pos=CC>and</wf>
<wf cmd=done pos=NN lemma=thanks wnsn=1 lexsn=1:10:00::>thanks</wf>
<wf cmd=ignore pos=IN>of</wf>
<wf cmd=ignore pos=DT>the</wf>
<wf cmd=done rdf=location pos=NNP lemma=location wnsn=1 lexsn=1:03:00:: pn=location>City_of_Atlanta</wf>
<punc>"</punc>
<wf cmd=ignore pos=IN>for</wf>
<wf cmd=ignore pos=DT>the</wf>
<wf cmd=done pos=NN lemma=manner wnsn=1 lexsn=1:07:02::>manner</wf>
<wf cmd=done pos=RB ot=notag>in</wf>
<wf cmd=done pos=RB ot=notag>which</wf>
<wf cmd=ignore pos=DT>the</wf>
<wf cmd=done pos=NN lemma=election wnsn=1 lexsn=1:04:01::>election</wf>
<wf cmd=done pos=VBD ot=notag>was</wf>
<wf cmd=done pos=VB lemma=conduct wnsn=1 lexsn=2:41:00::>conducted</wf>
<punc>.</punc>
</s>
</p>

Appendix B

Code for Implementation

Appendix 1 Source code for Graph Centrality Algorithms

```
public void Indegree()
{Iterator <String>it_comp=this.FinalGraph.Competing_vertices.iterator();
  while(it_comp.hasNext())
            {String label=it_comp.next();
              Vertex V=this.FinalGraph.MapSenseToVertex.get(label);
              List<Edge> Edges=this.FinalGraph.MapVertexToInGoingEdgeList.get(V);
              float sum=0f;
             Iterator <Edge> it=Edges.iterator();
             while(it.hasNext())
                        {   it.next();
                            sum++;
                        }
               V.score=sum;
            }
}
```

```
public void WeightedIndegree()
{Iterator <String>it_comp=this.FinalGraph.Competing_vertices.iterator();
  while(it_comp.hasNext())
            {String label=it_comp.next();
              Vertex V=this.FinalGraph.MapSenseToVertex.get(label);
              List<Edge> Edges=this.FinalGraph.MapVertexToInGoingEdgeList.get(V);
              float sum=0f;
             Iterator <Edge> it=Edges.iterator();
             while(it.hasNext())
                        {   sum+=it.next().weight;
                        }
               V.score=sum;
            }
}
```

```
public void PageRank()
{  int k=0;
    List<Vertex> Vertices=new ArrayList<Vertex>();
   Vertices.addAll(this.FinalGraph.Vertices);
    while(true)
            { k++;
              for(int i=0;i<this.FinalGraph.Vertices.size();i++)
            { Vertex Va=this.FinalGraph.Vertices.get(i);
                if (k==1) Va.score=1;
               List<Edge> InEdgeVa=FinalGraph.MapVertexToInGoingEdgeList.get(Va);
               float sumVba=0f;
               for(int j=0;j<InEdgeVa.size();j++)
                        {Vertex Vb=InEdgeVa.get(j).v1;
                          Vertex V2=InEdgeVa.get(j).v2;
                         Vb=FinalGraph.MapSenseToVertex.get(Vb.label);
                        V2=FinalGraph.MapSenseToVertex.get(V2.label);

            sumVba+=Vb.score/FinalGraph.MapVertexToOutGoingEdgeList.get(Vb).size();
                        }
                        double score=Va.score;
                        Va.score=1-0.85f+0.85f*sumVba;
                        }
            Vertices=new ArrayList<Vertex>();
            Vertices.addAll(this.FinalGraph.Vertices);
            if (k==20) break;
          }
      }

public void WeightedPageRank()
{ int k=0;
  List<Vertex> Vertices=new ArrayList<Vertex>();
  Vertices.addAll(this.FinalGraph.Vertices);
  while(true)
    { k++;
     for(int i=0;i<this.FinalGraph.Vertices.size();i++)
            {
               Vertex Va=this.FinalGraph.Vertices.get(i);
               if (k==1) Va.score=1;
               List<Edge> InEdgeVa=FinalGraph.MapVertexToInGoingEdgeList.get(Va);
               if (InEdgeVa==null) continue;
               float sumVba=0f;
            for(int j=0;j<InEdgeVa.size();j++)
            {Vertex Vb=InEdgeVa.get(j).v1;
              Vertex V2=InEdgeVa.get(j).v2;
              float sumVbc=0f;
            List<Edge> OutEdgeVb=null;
            OutEdgeVb=FinalGraph.MapVertexToOutGoingEdgeList.get(Vb);
            if (OutEdgeVb==null) continue;
            for(int l=0;l<OutEdgeVb.size();l++)
                    {  Edge bc=OutEdgeVb.get(l);
                       sumVbc+=bc.weight;
                    }
            Edge inEdgeVa=InEdgeVa.get(j);
            sumVba+=(inEdgeVa.weight/sumVbc)*Vb.score;
            }
            double score=Va.score;
            Va.score=1-0.85f+0.85f*sumVba;
            }
            Vertices=new ArrayList<Vertex>();
            Vertices.addAll(this.FinalGraph.Vertices);
            if (k==20) break;
          }
    }

public void HITS()
{Map<String,Float> Hub =new HashMap<String,Float>();
  int iteration=0;
  while(true)
 { iteration++;
    float auth_sum=0,hub_sum=0;
    for(int i=0;i<this.FinalGraph.Vertices.size();i++)
       { float auth=0f,hub=0f;
          Vertex V=this.FinalGraph.Vertices.get(i);
          String label=V.label;
            if (iteration==1)
            {
              V.score=1;
               Hub.put(label, 1f);
            }

            Iterator<Edge>
it=this.FinalGraph.MapVertexToInGoingEdgeList.get(V).iterator();
```

```
            while(it.hasNext())
            {  Edge edge=it.next();
               if (Hub.get(edge.v1.label)==null) Hub.put(edge.v1.label, 1f);
               auth+=Hub.get(edge.v1.label);
            }
            V.score=auth;
            auth_sum+=auth;
            it=this.FinalGraph.MapVertexToOutGoingEdgeList.get(V).iterator();
            while(it.hasNext())
            {  Edge edge=it.next();
               hub+=edge.v1.score;
            }
            hub_sum+=hub;

            Hub.put(V.label, hub);
            }
            for(int i=0;i<this.FinalGraph.Vertices.size();i++)
              { Vertex V=this.FinalGraph.Vertices.get(i);
                String label=V.label;
                V.score=V.score/auth_sum;
                Hub.put(label,Hub.get(label)/hub_sum);
              }
             if (iteration==20) break;
              }
            }


public void WeightedHITS()
{Map<String,Float> Hub =new HashMap<String,Float>();
  int iteration=0;
  while(true)
 { iteration++;
    float auth_sum=0,hub_sum=0;
    for(int i=0;i<this.FinalGraph.Vertices.size();i++)
       { float auth=0f,hub=0f;
          Vertex V=this.FinalGraph.Vertices.get(i);
          String label=V.label;
            if (iteration==1)
            {
              V.score=1;
               Hub.put(label, 1f);
            }

            Iterator<Edge>
    it=this.FinalGraph.MapVertexToInGoingEdgeList.get(V).iterator();
            while(it.hasNext())
            {  Edge edge=it.next();
               if (Hub.get(edge.v1.label)==null) Hub.put(edge.v1.label, 1f);
               auth+=edge.weight*Hub.get(edge.v1.label);

            }
            V.score=auth;
            auth_sum+=auth;
            it=this.FinalGraph.MapVertexToOutGoingEdgeList.get(V).iterator();
            while(it.hasNext())
            {  Edge edge=it.next();
               hub+=edge.weight*edge.v1.score;

            }
            hub_sum+=hub;

            Hub.put(V.label, hub);
            }
            for(int i=0;i<this.FinalGraph.Vertices.size();i++)
              { Vertex V=this.FinalGraph.Vertices.get(i);
                String label=V.label;
                V.score=V.score/auth_sum;
                Hub.put(label,Hub.get(label)/hub_sum);
              }
             if (iteration==20) break;
              }
            }
```

Appendix 2 Source code to assign ETF-INF edge weight in Graph

```
public void assign_ETFINF()
{for(int i=0;i<this.Vertices.size();i++)
  {Vertex V=this.Vertices.get(i);
   Map<String,Integer> EdgeTypeCount=new HashMap <String,Integer>();
   List <Edge>VEdges=this.MapVertexToOutGoingEdgeList.get(V);
   Iterator <Edge> itOutgoing=VEdges.iterator();
   while(itOutgoing.hasNext())
              {Edge edge=itOutgoing.next();
                     if (EdgeTypeCount.get(edge.edgeType)==null)
                     EdgeTypeCount.put(edge.edgeType, 1);
                     else EdgeTypeCount.put(edge.edgeType,
dgeTypeCount.get(edge.edgeType)+1);
               }
    itOutgoing=VEdges.iterator();
    while(itOutgoing.hasNext())
    { Edge edge=itOutgoing.next();
      ETF.put(edge, ((float)EdgeTypeCount.get(edge.edgeType)/VEdges.size()));
    }
Set <String> EdgeTypes= EdgeTypeCount.keySet();
  Iterator<String> it=EdgeTypes.iterator();
   while(it.hasNext())
   {    String edgetype=it.next();
     if (NF.keySet().contains(edgetype)) NF.put(edgetype, NF.get(edgetype)+1);
        else NF.put(edgetype, 1);
   }
}
Iterator<Edge> it=this.Edges.iterator();
while(it.hasNext())
{  try{   Edge edge=it.next();
edge.weight=ETF.get(edge)*(float)Math.log((((float)this.Vertices.size()+1)/NF.get(edge.e
dgeType));
 }
   catch(NullPointerException ex) {continue;}
  }
}
```

Appendix C

Outputs generated by the Implemented System

1. Input Sentence: "The fisherman jumped off the bank and into the water" Output: The fisherman#NN#1 jumped#VB#10 off the bank#NN#1 and into the water#NN#6

(Note: fisherman#NN#1 represents 1st sense assigned to noun fisherman by the system using WordNet 3.0 as sense inventory)

Meanings according to WordNet 3.0 :

fisherman#NN#1: someone whose occupation is catching fish

jump#VB#10: jump from an airplane and descend with a parachute

bank#NN#1: sloping land (especially the slope beside a body of water)

water#NN#6: a liquid necessary for the life of most animals and plants

2. Input Sentence: "He cashed a check at the bank"

Output: He cashed#VB#1 a check#NN#1 at the bank#NN#2

cash#VB#1: exchange for cash

check#NN#1: a written order directing a bank to pay money

bank#NN#2: a financial institution that accepts deposits and channels the money into lending activities

3. Input Sentence: "The student was grilled for two hours in exam"

Output: The student#NN#1 was grilled#VB#2 for two hours#NN#2 in exam#NN#1 student#NN#1: a learner who is enrolled in an educational institution grill#VB#2: examine thoroughly

hour#NN#2: clock time

exam#NN#1: a set of questions or exercises evaluating skill or knowledge

4. Input Sentence: “He grilled the sausages”

Output: He grilled#VB#1 the sausages#NN#1

grill#VB#1: cook over a grill

sausage#NN#1: highly seasoned minced meat stuffed in casings