Research on a Graph-based Method for Word sense Induction

By YIN BO

A thesis submitted to
School of Information Science,
Japan Advanced Institute of Science and Technology,
in partial fulfillment of the requirements
for the degree of
Master of Information Science
Graduate Program in Information Science

Written under the direction of
Associate Professor Kiyoaki Shirai

September, 2014

Research on a Graph-based Method for Word sense Induction

By YIN BO (1210211)

A thesis submitted to
School of Information Science,
Japan Advanced Institute of Science and Technology,
in partial fulfillment of the requirements
for the degree of
Master of Information Science
Graduate Program in Information Science

Written under the direction of
Associate Professor Kiyoaki Shirai

and approved by
Associate Professor Kiyoaki Shirai
Professor Satoshi Tojo
Associate Professor Minh Le Nguyen

August, 2014 (Submitted)

Contents

1 Introduction **1**
1.1 Background . . . 1
1.2 Goal . . . 2
1.3 Overview of the thesis . . . 3

2 Related Work **4**
2.1 Clustering-based Methods . . . 4
2.2 Graph based Methods . . . 5

3 Introduction to HyperLex **8**
3.1 Building Co-occurrence Graph . . . 8
3.1.1 Creating the Initial Graph . . . 8
3.1.2 Weighting in Co-occurrence Graph . . . 10
3.2 Detection of Components . . . 12
3.2.1 Detection of Root Hubs . . . 12
3.3 Delineating Components . . . 14
3.3.1 Minimum Spanning Tree (MST) . . . 14
3.3.2 Dividing Graph by MST . . . 15
3.4 Disambiguation . . . 17

4 Proposed Method **21**
4.1 Optimization of Parameters . . . 21
4.1.1 Introduction of "Small World Graph" . . . 22
4.1.2 Tuning Parameters Considering Small World Properties . . . 23
4.2 Finding Hubs . . . 26
4.2.1 Key Player Problem as Graph Connectivity Measure . . . 26
4.2.2 Finding the Hubs using *KPP* . . . 29
4.3 Weighting Edges Considering Syntactic Relations . . . 31
4.3.1 Syntactic Analysis with Standford Parser . . . 31
4.3.2 Weighting Scheme . . . 34

5 Evaluation **36**
5.1 Data . . . 36
5.2 Evaluation Measure . . . 39

5.3 Setting of Experiment 40
5.3.1 Range of Parameters 40
5.3.2 Threshold 41
5.4 Result of Baseline 41
5.5 Results of the Proposed Method 44
5.5.1 Results of Optimization of Parameters 44
5.5.2 Results of Detection of Components Using KPP 47
5.5.3 Results of Weighting Scheme Considering Syntactic Relation 51

6 Conclusion **55**
6.1 Summary 55
6.2 Future Work 56

Acknowledge **57**

Bibliography **58**

List of Figures

3.1 Co-occurrence Graph of Target Word 'activate' 10
3.2 Example of Selection of a Hub 13
3.3 Algorithm of Finding the Hubs 13
3.4 MST of Target Word 'different' 16
3.5 Algorithm of Delineating Components 16
3.6 Instance of 'ask' (1) 18
3.7 Instance of 'ask' (2) 18

4.1 Algorithm of Parameter Optimization 25
4.2 Flowchart of Choosing the Best Parameter Combination 26
4.3 An Example of a Graph 27
4.4 Example of Isolated Candidate of the Hub 30
4.5 Algorithm for Detecting Hubs using *KPP* 31
4.6 Basic Dependencies of a Sentence 35

5.1 Distribution of the Size of Corpora 38
5.2 Results of *HyperLex_OP* for Different T_2. 46
5.3 Results of all *HyperLex_OP+KPP* 50
5.4 Results of all *HyperLex_OP+KPP+SYN* 53

List of Tables

3.1 Number of Co-occurrences of Two Words 11
3.2 Nine components of ‘note’ 14
3.3 Score Vector for the Instance *ask.v.bnc.00006994* 19
3.4 Score Vector for the Instance *ask.v.bnc.00008747* 20

4.1 Labels of Grammatical Relations in Stanford Parser 32
4.2 Example Sentence and its Results of POS Tagging 33
4.3 List of Grammatical Dependencies for the Example Text 34

5.1 Number of Instances for Each Target Word 37
5.2 Range of Parameters 41
5.3 Range of Thresholds 41
5.4 Results of *HyperLex* for All Target Words (1) 42
5.5 Results of *HyperLex* for All Target Words (2) 43
5.6 Results of *HyperLex_OP* for All Target Words (1) 44
5.7 Results of *HyperLex_OP* for All Target Words (2) 45
5.8 Number of Target Words for which No Graph is Obtained 46
5.9 Comparison of Baseline and *HyperLex_OP* 47
5.10 Results of *HyperLex_OP+KPP* for All Target Words (1) 48
5.11 Results of *HyperLex_OP+KPP* for All Target Words (2) 49
5.12 Comparison among Baseline, *HyperLex_OP* and *HyperLex_OP+KPP* . . . 50
5.13 Results of *HyperLex_OP+KPP+SYN* for All Target Words (1) 51
5.14 Results of *HyperLex_OP+KPP+SYN* for All Target Words (2) 52
5.15 Comparison of *HyperLex_OP+KPP+SYN* and Other Systems 53

Chapter 1

Introduction

1.1 Background

In computational linguistics, Word Sense Induction (WSI) is an open problem of natural language processing, which concerns automatic identification of senses of a word. The senses of words are usually defined by a dictionary, but it might not compile all senses, especially a new sense or a sense in a specific domain. Therefore, WSI is important to recognize all the senses for a given word and indispensable for various NLP applications, such as machine translation, information retrieval, text-to-speech synthesis and so on.

WSI is a task to construct clusters of contexts in which the target word occurs or to build cluster of words related to the sense of the target word. Each obtained cluster corresponds to one sense of the target word. In order to solve WSI problem, two major approaches have been proposed, clustering based method and graph based method.

Clustering based method

Words are semantically similar if they appear in similar documents, in similar context windows, or in similar syntactic contexts. Based on this idea, each occurrence of a target word in a corpus is represented as a context vector. The contexts of the target word are similar if many words in the contexts are overlapped. The vectors are then clustered into groups. Finally, each cluster represents a sense of the target word.

Graph based method

Words are represented by a co-occurrence graph, whose vertices are words appearing in the context of the target word and edges are co-occurrence relations. Then the graph is divided into several sub-graphs. Each sub-graph represents an induced sense of the target word.

As we will discuss in Chapter 2, many papers have been devoted to the study of WSI. However, there are much room to improve the WSI methods.

1.2 Goal

HyperLex is one of the graph-based word sense induction method proposed by Véronis [1]. As usual graph based WSI methods, HyperLex infers senses of a word in the following procedures. First, the graph called 'co-occurrence graph' is built. In the co-occurrence graph, words are represented as vertices, and edges are connected between words that frequently co-occur in the corpus. The goal of HyperLex is to divide the graph into several dense sub-graphs (or sub-trees) so that each sub-graph stands for a set of words appearing in the same context of one sense of the target word. To divide the graph, several nodes that might be a center of the dense sub-graph, called 'hub', are chosen. Then the graph is split into sub-trees whose root nodes are hubs. Roughly saying, one sub-tree represents one induced sense. Although Veronis reported promising results on HyperLex, there are some problems on it.

The first problem concerns the way how to construct the co-occurrence graph. In HyperLex, there are several parameters to decide words to be added as vertices of the graph, or pairs of words to be added as edges. Therefore, the structure of the co-occurrence graph highly depends on these parameters. However, parameters are determined in ad-hoc manner. Since the best parameter may be different for the target word or the corpus used for WSI, a scheme to determine the optimum parameters should be investigated.

The second problem concerns how to choose the hub. In HyperLex, the frequent words in the corpus are simply chosen as hubs. However, the structure of the graph should be considered when we choose the hubs, since the hub should be the center of the dense sub-graph.

The third problem concerns how to determine weights of edges. In a graph-based WSI, edges in the co-occurrence graph are weighted so that the weight is small when two words correlate each other. In HyperLex, only the co-occurrence frequency of two words is considered to set weights of the edges. However, correlation between two words can be measured from other points of views. Syntactic relation is one of them. If two words are under a syntactic relation more frequently in the corpus, their relations might be strong and the weight of the edge connecting them should be set small. Although it is well-known that the syntactic relation is one of the useful features for word sense disambiguation, it is not considered in HyperLex.

The goal of this thesis is to propose a method to tackle the above problems and improve the performance of HyperLex for WSI. Our goal can be summarized as follows:

- to propose a method for optimization of parameters
- to investigate a better way to find hubs
- to investigate a way to incorporate syntactic relations between two words into the weights of the edges in the co-occurrence graph

1.3 Overview of the thesis

The thesis is organized as follows. Chapter 2 introduces previous work on WSI. Chapter 3 presents the detail algorithm of HyperLex. The proposed method to improve the original HyperLex will be described in Chapter 4. Chapter 5 describes evaluation of the proposed method. Finally, the conclusion and future work is presented in Chapter 6.

Chapter 2

Related Work

As discussed in Chapter 1, there are two approaches for WSI: clustering based method and graph based method. Following sections introduce previous work of two approaches.

2.1 Clustering-based Methods

In a clustering based method, all characteristics of a target word in a corpus are extracted as a context vector. The characteristics mainly contain the co-occurrence words, parts of speech, topic characteristics, N-grams, phrases, and so on. Then the senses of a target word is induced by clustering algorithm.

Pinto et al. described a simple technique for unsupervised sense induction for ambiguous words [2]. The method used the point wise mutual information for calculating a set of co-occurrence terms which were used to expand the original context vector. Once the expansion has been done, the unsupervised KStar clustering method was used to induce the sense of each ambiguous word.

Purandare and Pedersen systematically compared several unsupervised word sense discrimination techniques [3]. Four kinds of characteristics of the WSI methods were presented: clustering space, context vector, feature of context vector and clustering algorithm. As for clustering space, vector and similarity space were considered. In the vector space, a dimension of a context vector of an instance was a word appearing in the context. On the other hand, in the similarity space, a dimension of a context vector was other instances, and a value for each dimension was similarity between instances. As for the context vector, the first and second order context vectors were considered. The first order context vector was made directly from the context of the target word, while the second order context vector was made by summing the first order context vector of surrounding words. Next, two kinds of features of the context vector were considered: uni-gram and bi-gram of words. Finally, hierarchical, partitional and hybrid of them were considered as clustering algorithm. Several WSI methods were implemented by combining the above 4 characteristics. They were empirically compared in the paper.

Elshamy et al. used topic features to cluster different word senses in their global context topic space [4]. Using unlabeled data, this system trained a latent dirichlet allocation (LDA) topic model then used it to infer the topic distribution of the test instances. The instances of the target words were merged into several clusters according to the similarity of the topic distribution. Their hypothesis was that closeness in topic space reflects similarity between different word senses.

Cruys presented an extension of a dimensionality reduction algorithm called NONNEGATIVE MATRIX FACTORIZATION (NMF), which combined both 'bag of words' data and syntactic data to find semantic dimensions according to which both words and syntactic relations can be classified [5]. They applied NMF to the three matrices. The first matrix contained co-occurrence frequencies of nouns cross-classified by dependency relations. The second matrix contained co-occurrence frequencies of nouns cross-classified by words that appear in the noun's context window, and the third matrix contained co-occurrence frequencies of dependency relations cross-classified by co-occurring context words. The use of three matrices allowed one to determine which dimensions were responsible for a certain sense of a word, and adapted the corresponding feature vector accordingly, 'subtracting' one sense to discover another one. The extended NMF was embedded in existing clustering frameworks for word sense induction.

2.2 Graph based Methods

In graph based methods, elements and characteristics of clustering are defined in the graph, then the senses are induced by using the graph based clustering algorithms. In the graph, nodes can represent words in the context of the target word, or can represent the instances[1] of the target word, and edges represent the similarity between the nodes.

Véronis proposed the HyperLex algorithm using the co-occurrence graph [1]. Pairs of words in the context were built in the form of co-occurrence graphs. A node in the graph was a word appearing in the context of the target word. An edge was used to connect two nodes that co-occurred in the same sentence. Next, the nodes connected with many other nodes, called hubs, were identified. Finally, the graph was divided into several sub-trees whose root node was the hub. Each sub-trees represented an induced sense of the target

[1]sentences including the target word.

word. That is, the words in one sub-tree were words appearing in the context of the one sense of the target word.

Agirre et. al. explored two sides of HyperLex: the optimization of 7 parameters in HyperLex and the empirical comparison on a standard benchmark against other WSD systems [6]. They used hand-tagged corpora to map the induced senses to WordNet senses for comparison among WSD systems. The method allowed to fine-tune the parameters automatically using another sense tagged corpus as a development data. However, it requires much human labor to construct a sense tagged corpus for parameter tuning. This thesis also proposes a method for parameter optimization of HyperLex. Our method does not utilize hand-tagged sentences but choose the best parameters according to a structure of a co-occurrence graph.

Agirre and Soroa proposed a graph-based unsupervised system for induction [7]. The way of building a graph was similar to HyperLex. For each target word, a graph consisting of words in the context was built. Vertices in the co-occurrence graph were words and two vertices shared an edge whenever they co-occured in the same context. Besides, each edge received a weight, which indicated how strong the nodes were related each other. Then PageRank algorithm was used to identify the hubs in the graph. Each context (a sentence including the target word) was represented by a hub score vector. Finally, clusters of contexts were constructed by Markov clustering algorithm.

Dorow and Widdows proposed an algorithm which automatically discovered word senses from free text and mapped them to the appropriate entries of existing dictionaries or taxonomies [8]. The algorithm was based on a graph model representing words and relationships between them. They assumed a co-occurrence word corresponded to a sense of polysemous word, and used co-occurrence nouns (nouns occurring near the target word) as nodes of the graph. If the number of co-occurrence of two nouns were more than a given threshold, an edge between these two nodes was added. The weight of the edge was set to the number of co-occurrence of them. Then, Markov clustering (MCL) algorithm was used to construct clusters of contexts. Sense clusters were iteratively computed by clustering the local graph of similar words around the target word. Discrimination against previously extracted sense clusters enabled us to discover new senses. Finally, they used the same data for both recognizing and resolving ambiguity of the senses.

Manandha and Klapaftis assumed that the collocation of two words corresponded to a sense of the polysemous word [9]. The use of collocations instead of single words can alleviate the problem of word sense induction. In the algorithm, the collocations represented as nodes in the graph, and the weight in the edges was set up by a conditional probability of the presence of the collocations. As the graph which was built by this method is very sparse [10], the authors considered that two nodes in the graph which were similar to each other should have the same neighbors, and obtain more edges by finding the nodes, so that the smooth collocational graph was constructed. Finally, clusters of contexts were constructed by a clustering algorithm called “Chinese Whispers” (CW) [11].

As the graph using collocations as nodes led to sparseness, Klapaftis and Manandhar proposed an unsupervised graph based method for automatic word sense induction and disambiguation to solve this problem [12]. Their approach represented an unambiguous unit as a vertex in a graph. The unambiguous unit was either a single word if it was judged as unambiguous or a pair of words if not. Graph edges modeled the co-occurrences of the content of the vertices. Hard-clustering of the graph induced a set of senses. In the disambiguation stage, each induced cluster was scored according to the number of its vertices found in the context of the target word.

Jurgens proposed a new graph-based method for WSI based on finding sense-specific word communities within a co-occurrence graph [13]. This approach had two features: (1) the impact of word frequency on community size and memberships and (2) identifying both graph properties and semantic relations within hierarchical communities that distinguish between sense granularities. He also proposed a method to distinguish the induced sense of the target word in a new context. Software for the WSI model and link clustering is available as a part of the S-Space Package.

Lau applied topic modeling to automatically induce word senses of a target word, and demonstrated that their word sense induction method can be used to automatically detect words with emergent novel senses, as well as token occurrences of those senses [14]. They explored a WSI method based on the utility of standard topic models, with a pre-determined number of topics (=senses). This paper next demonstrated that a non-parametric formulation that learned an appropriate number of senses per a word actually performed better at the WSI task.

In the graph based methods, the selection of system parameters has an important influence to its performance, and the parameters are generally set up based on experience or supervised learning. Korkontzelos et al. focused on the unsupervised estimation of the free parameters of a graph-based WSI method, and explored the use of eight Graph Connectivity Measures (GCM) that assessed the degree of connectivity in a graph [15]. Eight graph connectivity measures used to find the parameters were Average Degree, Average Weighted Degree, Average Cluster Coeficient, Average Weighted Cluster Coeficient, Graph Entropy, Weighted Graph Entropy, Edge Density and Weighted Edge Density. Given a target word and a set of parameters, GCM evaluated the connectivity of the produced clusters, which corresponded to sub-graphs of the initial (unclustered) graph. Each parameter setting was assigned a score according to one of the GCM and then the highest scoring setting was selected.

Chapter 3

Introduction to HyperLex

HyperLex [1] is one of the excellent graph-based WSI methods. The proposed method in this thesis is modification of HyperLex to overcome several problems in it. Before describing our proposal, this chapter introduces the detail algorithm of HyperLex.

The goal of Word Sense Induction is to guess senses of a certain word, called 'target word'. HyperLex accepts a collection of sentences including the target word as an input. Then it induces the senses of the target words as follows: (1) a co-occurrence graph whose nodes are words in the context of the target word is constructed, (2) the nodes which represents the sense, called 'hub', are identified, (3) the graph is subdivided into several sub-trees whose root node are hub, (4) and finally a sense of a word in a new context is disambiguated using the sub-trees. The detail of each step will be described in the following sections.

3.1 Building Co-occurrence Graph

For each target word, a co-occurrence graph consisting of words appearing in the context of the target word is built. Véronis shows that this kind of a graph fulfills the properties of small world graphs, and thus possess highly connected components. The centers or prototypes of these components, called hubs, eventually identify the main word uses (senses) of the target word.

3.1.1 Creating the Initial Graph

We start to build an initially-empty undirected graph $G = (V, E)$, where $V = E = \varnothing$.

The graph is constructed from a corpus consisting of sentences including the target word. At first, sentences containing fewer than 4 words were deleted from the corpus. The threshold for removing the short sentences is called a parameter P_1 in this thesis. (i.e. P_1 is set to 4 in HyperLex). A co-occurrence graph for the target word is built by the following procedures.

Graph vertices (V): Words that occur less than 10 times in the entire corpus are discarded. The threshold of the frequency of the word is called a parameter P_2 in this thesis (i.e. P_2 is set to 10 in HyperLex). The remaining words become vertices in a graph G.

Graph edges (E): It is said that two words 'co-occur' if they appear in the same sentence in the corpus. If two vertices co-occur more than 5 times, they are connected with an edge. The threshold of the number of co-occurrence for making the edge between two words is called a parameter P_3 in this thesis (i.e. P_3 is set to 5 in HyperLex).

Finally, we have obtained the initial co-occurrence graph. Figure 3.1 shows the co-occurrence graph of the target word 'activate'. This example graph is obtained from the corpus used for the evaluation. The details of the corpus will be described in Chapter 5.

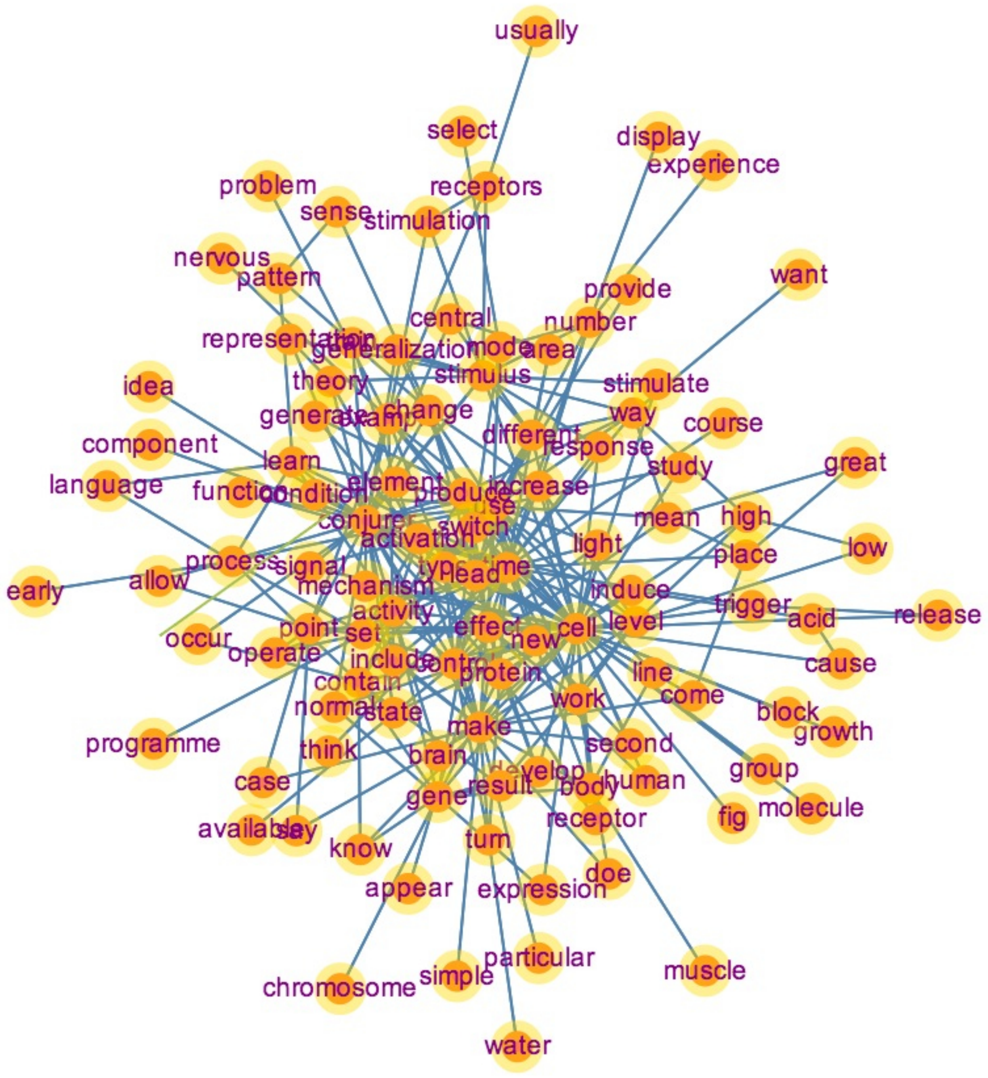


Figure 3.1: Co-occurrence Graph of Target Word ‘activate’

3.1.2 Weighting in Co-occurrence Graph

Each edge is assigned with a weight which measures the relative frequency of the co-occurrence of two words. Let w_{ij} be the weight of the edge connecting vertices n_i and n_j. w_{ij} is defined by Equation (3.1):

$$w_{ij} = 1 - \max \left\langle p\left(n_i \middle| \, n_j\right) \;, p\left(n_j \middle| \, n_i\right) \right\rangle \tag{3.1}$$

In (3.1), $p(n_i|n_j)$ is the conditional probability of observing n_i in a given context, knowing that that context contains n_j, and inversely, $p(n_j|n_i)$ is the probability of observing n_j in a given context, knowing that it contains n_i. These probabilities are estimated from frequencies in the corpus as Equation (3.2).

$$p\left(n_i \middle| \, n_j\right) = \frac{freq\left(n_i, n_j\right)}{freq\left(n_j\right)} \qquad\qquad p\left(n_j \middle| \, n_i\right) = \frac{freq\left(n_i, n_j\right)}{freq\left(n_i\right)} \tag{3.2}$$

For example, let us consider how to calculate the weight between 'know' and 'make', and the weight between 'different' and 'make'. Table 3.2 shows the number of contexts in both or each or none of words appear.

	know	∼know	Total		make	∼make	Total
make	5	49	54	different	0	35	35
∼make	14	844	858	∼different	54	823	877
Total	19	893	912	Total	54	858	912

Table 3.1: Number of Co-occurrences of Two Words

From the statistics in Table 3.2, the weights are calculated as follows:

$$p\left(know \,|make\right) = 5/54 = 0.0926 \qquad p\left(make \,|know\right) = 5/19 = 0.2632$$

$$p\left(make \,|different\right) = 0/35 = 0 \qquad p\left(different \,|make\right) = 0/54 = 0$$

$$w\left(know\text{-}make\right) = 1 - \max\left(0.0926\,,\, 0.2632\right) = 0.7368$$

$$w\left(different\text{-}make\right) = 1 - \max\left(0\,,\, 0\right) = 1$$

The weight of an edge reflects the magnitude of the semantic distance between words: words which always occur together receive a weight of 0. Rarely co-occurring words receive a weight close to 1.

3.2 Detection of Components

Once the co-occurrence graph is built, a simple iterative algorithm is applied to obtain its components. The basic assumption underlying the simple iterative algorithm is that the different senses of a target word form highly interconnected 'bundles' in a small world of co-occurrences, or in terms of graph theory, high-density components. For example, the target french word 'barrage' have to co-occur frequently with 'water', 'work', 'river', 'flood', 'irrigation', 'production', 'electricity', etc.[1], and these words themselves are likely to be highly interconnected. So we can call these word as one bundle (component). In addition, the target word 'barrage' have to co-occur frequently with 'match', 'team', 'cup', 'world', 'soccer', 'victory', etc., and these words themselves also are likely to be highly interconnected. Similarly, we can call these words as the other bundle (component). Intuitively, the first bundle corresponds to 'dam' sense of 'barrage', while the second corresponds to 'play-off' sense.

Since, detecting the high-density components in the co-occurrence graph is NP-hard unfortunately, HyperLex detects the components by application specific heuristics. The detail algorithm to detect the components in the graph will be described in the rest of this section.

3.2.1 Detection of Root Hubs

In every high-density component, one of the nodes has a higher degree than the others. Such a node is called the root hub of the component. All the hub nodes are identified iteratively. For example, let us suppose that Figure 3.2 is the co-occurrence graph of the target word 'different'. The word 'art' is chosen as the first hub because it is the most frequent word in the graph. Actually, the degree of the node 'art' (the number of edges connected to 'art') is high. So the heuristic can choose the appropriate hub in this case. After choosing the hub, we deleted the root hub along with all of its neighbors. Figure 3.2 also shows the hub and its neighboring nodes to be deleted.

[1]These are actually French words. English translations are written in the thesis for better readability. Note that this example is introduced in the paper [1].

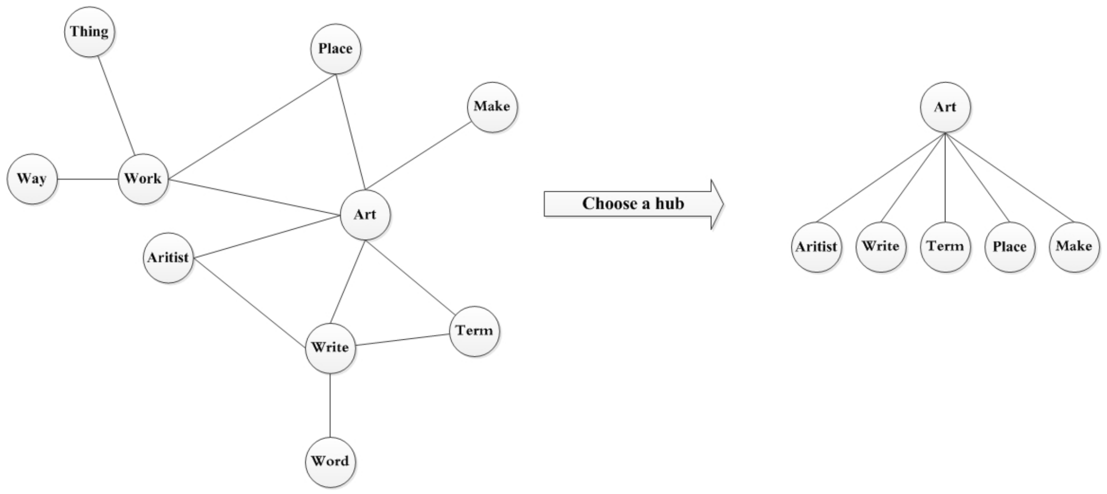


Figure 3.2: Example of Selection of a Hub

Then the root hub of the next component is identified. The most frequent node in the remained graph is chosen as the next hub. HyperLex continues this process until no node is left.

Note that, instead of going through the nodes in decreasing order of the degree which requires heavy computations at the node level, they are scanned in decreasing order of frequency. It is supposed that the genuine degree and the frequency of the node are highly correlated, and the selection of the hubs according to the frequency might be good approximation.

A pseudocode of the algorithm for finding the hubs can be written in Figure 3.3:

```
Root Hubs($G$) {
    $G$: co-occurrence graph;
    $H$: a set of hubs;
    $V \leftarrow$ nodes in $G$ sorted in descending order of frequency;
    $H \leftarrow \varnothing$
    While ($V \neq \varnothing$) {
        $v \leftarrow$ the first node in $V$;
        $H \leftarrow H \cup \{v\}$;
        $N \leftarrow$ neighbors of $v$ and $v$ itself;
        $V \leftarrow V \setminus N$;
    Return $H$; }
}
```

Figure 3.3: Algorithm of Finding the Hubs

Table 3.2 shows hubs and its neighbors of a target word 'note'. This example is excerpted from the results of the experiment described in Chapter 5.

Hub	**Neighbors of Hubs**
Time	British, Mean
Conjurer	Include, Conjurer, Form, Member, Case
Service	Local, Form, Appear, Leave
Come	Particularly, Public
Church	Good
Social	People, Use
Year	Change, New
high	Make
particular	Group, State

Table 3.2: Nine components of 'note'

3.3 Delineating Components

3.3.1 Minimum Spanning Tree (MST)

Given a connected, undirected graph, a spanning tree of that graph is a sub-graph that is a tree and connects all the vertices together. A single graph can have many different spanning trees. We can also assign a weight to each edge, which is a value representing how unfavorable it is. Now the score of the spanning tree is defined as the sum of the weights of the edges in it. A minimum spanning tree (MST) or minimum weight spanning tree is a spanning tree whose score (sum of the weights) is minimum. More generally, any undirected graph (not necessarily connected) has a minimum spanning forest, which is a union of minimum spanning trees for its connected components.

Prim's algorithm: This is a greedy algorithm that finds a minimum spanning tree for a connected weighted undirected graph. It finds a subset of the edges that forms a tree containing every vertex, where the total weights of all the edges in the tree is minimized. The procedure of Prim's algorithm is as follows:

A. Input: A non-empty connected weighted graph with vertices V and edges E (the weights can be negative).

B. Initialize: $V_{new} = \{x\}$, where x is an arbitrary node (starting point) in V, $E_{new} = \varnothing$.

C. Repeat until $V_{new} = V$:

  i. Choose an edge $\{u, v\}$ with minimal weight such that u is in V_{new} and v is not (if there are multiple edges with the same weight, any of them can be picked).

  ii. Add v to V_{new}, and $\{u, v\}$ to E_{new}.

D. Output: V_{new} and E_{new} describe a minimal spanning tree.

3.3.2 Dividing Graph by MST

Once the hubs that represent the senses of the word are selected, an expanded graph is built. First, a dummy node representing the target word itself is added to the graph. Then each hub is linked to the target word with 0 weights. To infer the sense of the target word, a minimum spanning tree (MST) is computed over the expanded graph by taking the target word as the starting node and making previously identified hubs as its first level children. Finally, the target word is removed, causing that the graph is subdivided into the sub-trees whose root nodes are hubs. Figure 3.4 shows an example of MST of the target word ‘different’. By removing the node ‘different’, the graph can be divided into 7 components. Now each component corresponds to an induced sense of the target word. That is, the nodes in the sub-tree represent words appearing in the context of the induced sense.

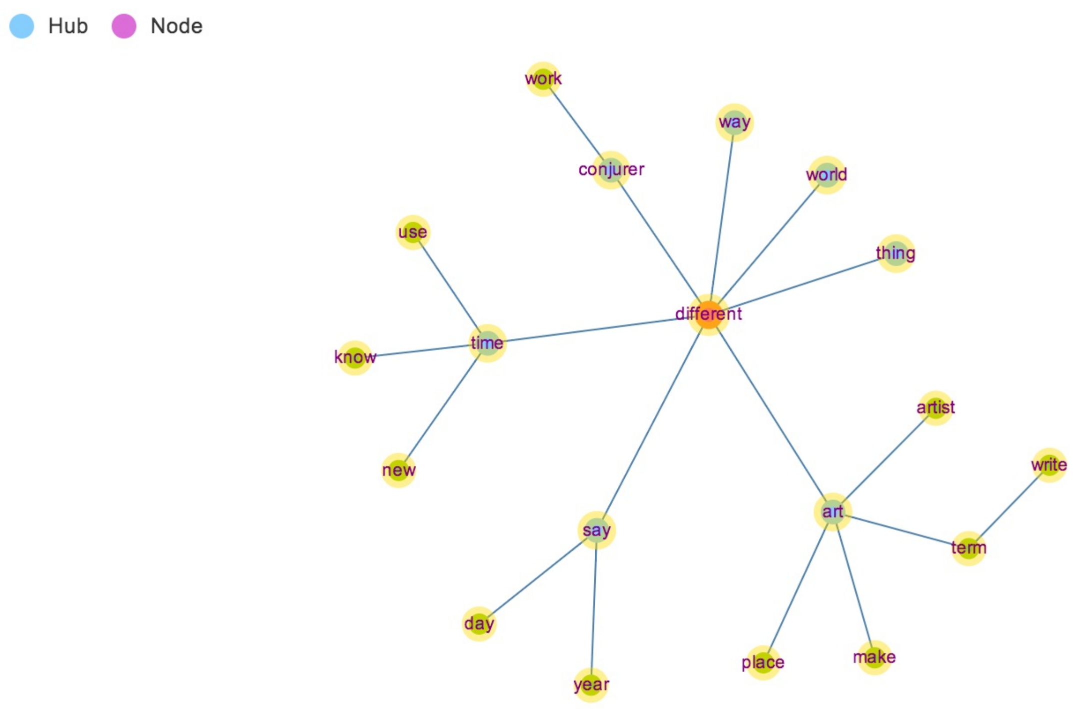


Figure 3.4: MST of Target Word 'different'

A pseudocode of this algorithm is described in Figure 3.5:

```
**Components**$(G, H, T)$ {
    $G$: co-occurrence graph;
    $N$: set of nodes in G;
    $E$: set of edges in G;
    $H$: set of root hub;
    $T$: target word;
    $N \leftarrow N \cup T$;
    for each $h$ in $H${
        Add an edge $< t, h >$ with weight 0 to $E$;
    }
    $G' \leftarrow$ an extended graph of $\{N, E\}$;
    $Tree \leftarrow MST < G', T >$;
    $C \leftarrow$ set of trees obtained by removing $T$ from $Tree$;
    Return $C$;
}
```

Figure 3.5: Algorithm of Delineating Components

3.4 Disambiguation

The components are then used to perform word sense disambiguation. For every instance of the target word, the words surrounding it are examined and confronted with the components.

Words in the co-occurrence graph are placed under exactly one component whose root node is a hub. Each word in the context receives a set of scores S_i of the hub h_i. The score S_i is calculated as in Equation (3.3)

$$S_i = \begin{cases} \frac{1}{1+d(h_i,v)} & if\ v\ belongs\ to\ components\ i \\ 0 & otherwise \end{cases} \tag{3.3}$$

, where $d(h_i, v)$ is the distance between root hub h_i and node v in the tree. Equation (3.3) 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 hub. Score vector consisting of S_i for all components is constructed. In the score vector, all values are 0 except for one score corresponds to the component to which the word belongs. For a given context, the score vectors of all words in the context is added, and the component that receives the highest score is chosen.

Figure 3.6 and Figure 3.7 shows two instances of the target word 'ask' excerpted from the corpus used for the evaluation in Chapter 5.

Instance of 'ask': ask.v.bnc.00008747

Nowadays, nearly every young person has probably had some experience of basic improvisation at their school or through the extensive TIE (Theatre in Education) tours. In drama schools, improvisation is about finding a way of expanding the imagination and liberating the senses, which can get too confined if students work entirely from a text all the time. The use of impro in training has gone through many phases; it still conjures up the traditional, hackneyed image of a student being **asked** to be a tree or an ice cream. But it is possible to go way beyond these limited, obvious exercises, and impro can be immensely exciting for young actors, allowing them to grasp situations and emotions imaginatively, perhaps for the first time. Here is an example of an impro exercise for two actors: An actor is asked to assume the character of a close family friend who arrives at the house with the news of the death of the wife ś husband in an accident.

Figure 3.6: Instance of 'ask' (1)

Instance of 'ask': ask.v.bnc.00006994

We will see that this mode of constructing monopoly Catholicism is alien to the Concordat period of Roman catholic church state relations recognizable in arrangements, for example, with Spain, Portugal, and Italy. A dislike by Irish clerics for such an explicit and direct church state relationship has a long tradition and comes out best in De Tocqueville 's (1957) conversations with Irish clergy on his visit to Ireland in 1834. When **asked** if they would like subventions from the state to aid their stipends and church buildings, a move which was being seriously considered by the British government at the time, priests and bishops were united in rejecting the idea on the grounds that it would drive a wedge between clergy and people, identifying clergy with the principal enemies of the people. De Tocquevilleś notes reveal not only the conscious opposition to such a mode of religious power but also how deep the solidarity between clergy and people was, the degree to which the poor, half the catholic population at the time, looked to the clergy for material and spiritual leadership, guidance, and assistance, and how much they trusted them. However, with the emergence of the Southern Irish state, it soon became clear that this secularization of the state form was not to signify an absence of Roman catholic power in the construction of public morality, but rather an indirect recognition of the sovereignty of the church in most areas of moral concern besides education.

Figure 3.7: Instance of 'ask' (2)

Table 3.3 and Table 3.4 show respectively the score vectors for the instances *ask.v.bnc.00006994* and *ask.v.bnc.00008747*. In these tables, 'Time', 'Man', 'Thing', 'Doctor' and 'Force' stand for the hubs. For example, for the instance *ask.v.bnc.00006994*, the word 'allow' belongs to the component 'Time' and the score for 'Time' is 0.82; its score vector is $(0.42, 0, 0, 0)$. Similarly, 'school' belongs to the component 'Man' and the score for 'Man' is 1.54; its score vector is $(0, 0.24, 0, 0)$. In addition, for the instance *ask.v.bnc.00008747*, 'best' belongs to the component 'Man' and the score for 'Man' is 0.28; its score vector is $(0, 0.28, 0, 0)$. Similarly, 'idea' belongs to the component 'Time' and the score for 'Time' is 0.62; its score vector is $(0.62, 0, 0, 0)$. The sum of the score vectors for all words in the context are obtained. Then the component with the highest score is chosen as the sense of the target word. For the instance *ask.v.bnc.00006994*, the component 'Time' is chosen since its score 3.68 is the highest. On the other hand, for the instance *ask.v.bnc.00008747*, the component 'Man' is selected as its score 3.42 is the highest.

ask.v.bnc.00006994					
Word	**Time**	**Man**	**Thing**	**Doctor**	**Force**
allow	0.42	0	0	0	0
drama	0	0.24	0	0	0
family	0.65	0	0	0	0
house	0.65	0	0	0	0
school	0	0.22	0	0	0
situation	0	0.30	0	0	0
student	0.40	0	0	0	0
time	1	0	0	0	0
train	0.56	0	0	0	0
use	0	0	0	0.6	0
way	0	0.36	0	0	0
work	0	0.27	0	0	0
young	0	0.6	0	0	0
Total	**3.68**	**1.99**	**0**	**0.6**	**0**

Table 3.3: Score Vector for the Instance *ask.v.bnc.00006994*

ask.v.bnc.00008747					
Word	**Time**	**Man**	**Thing**	**Doctor**	**Force**
best	0	0.24	0	0	0
come	0	0.48	0	0	0
conjurer	0.58	0	0	0	0
government	0.62	0	0	0	0
idea	0.62	0	0	0	0
long	0	0.37	0	0	0
look	0	0.35	0	0	0
people	0.58	0	0	0	0
time	1	0	0	0	0
Total	**3.39**	**3.42**	**0**	**0**	**0**

Table 3.4: Score Vector for the Instance *ask.v.bnc.00008747*

Chapter 4

Proposed Method

In this chapter, we will introduce the proposed methods based the algorithm of HyperLex. As already explained, our method is the revision of HyperLex in the following three ways: (1) optimization of parameters, (2) a method to choose the hubs and (3) weighting scheme of edges considering syntactic relations. The following sections explain our method one by one.

4.1 Optimization of Parameters

In this section, we investigate the ways to choose parameter values. There are three parameters in HyperLex, such as:

- A threshold of a length of a sentence to be used to make a graph (parameter P_1)
- A threshold of a frequency of a word to be added as a node (parameter P_2)
- A threshold of a co-occurrence frequency of a pair of words to be added as an edge (parameter P_3)

In HyperLex, these parameters are fixed. Since the structure of the co-occurrence graph is highly dependent on these parameters, however, they play an important role in WSI. It is essential to optimize the parameters to guess the senses of the target word more precisely.

To address the issue of choosing parameter values, we will consider all combinations of parameters so that the constructed graph meets the condition of small world properties.

4.1.1 Introduction of "Small World Graph"

To optimize the parameters, we have to consider the properties of the "Small World Graph" to build co-occurrence graph.

Véronis claims that the co-occurrence graph have the properties of the "Small World Graph" , which is proposed by Watts and Strogatz [16], one of today's key areas of research in graph theory. While a large part of the graph-theory studies have dealt with regular graphs or random graphs, Watts and Strogatz has shown that most real world graphs and networks do not fall into either of these categories, but are in an intermediate state somewhere between order and chaos. Roughly saying, "small world graph" stands for a graph where several dense subsets of nodes (small world) are loosely connected. For WSI, it is appropriate that the co-occurrence graph has a small world property, since the small world in the graph may stand for each sense. In other words, to infer the senses of the target word, it is necessary to construct a small world graph.

Watts and Strogatz defined two measures for characterizing small world graphs: the characteristic path length (L) and the clustering coefficient (C). L is the mean length of the shortest path between two nodes in the graph. Let $d_{min}(i,j)$ be the length of the shortest path between two nodes, i and j, and let N be the total number of node. L is defined as Equation (4.1). The characteristic path length represents "closeness" of nodes in the graph.

$$L = \frac{1}{N}\sum_{i=1}^{N} d_{\min}\left(i,j\right) \tag{4.1}$$

While the clustering coefficient of a graph shows the extent to which nodes tend to form connected groups that have many edges connecting each other in the group, and few edges leading out of the group. For each node i, one can define a local clustering coefficient C_i equal to the proportion of connection $E\left(\Gamma\left(i\right)\right)$ between the neighbors $\Gamma\left(i\right)$ of that node where E stands for a set of edges connecting nodes in a given set of nodes. For a node i with 4 neighbors, for instance, the maximum number of connections is $\begin{pmatrix} |\Gamma\left(i\right)| \\ 2 \end{pmatrix} = 6$. If 5 of these connections actually exist, $C_i = 5/6 \approx 0.83$. The global coefficient C is the mean of the local coefficient as shown in Equation (4.2).

$$C = \frac{1}{N}\sum_{i=1}^{N} \frac{|E\left(\Gamma\left(i\right)\right)|}{\begin{pmatrix} |\Gamma\left(i\right)| \\ 2 \end{pmatrix}} \tag{4.2}$$

This coefficient ranges between 0 for a totally disconnected graph and 1 for a complete graph. When the edges are weighted, a weighted clustering coefficient C_{weight} can be defined as in Equation (4.3).

$$C_{weight} = \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{|E(\Gamma(i))|} \frac{(1 - w_{ij})}{\binom{|\Gamma(i)|}{2}} \tag{4.3}$$

In the case of a random graph, Equation (4.4) is supposed to be fulfilled. N is a number of nodes in the random graph. k is a mean of number of edges per a node, or E/N,

$$L_{rand} \sim \frac{\log(N)}{\log(k)} \qquad\qquad C_{rand} \sim \frac{2k}{N} \tag{4.4}$$

For example, a random graph of 1000 nodes and 10000 edges will have a mean degree k of 10, the characteristic path length L_{rand} is $log(1000)/log(10) = 3$ and the clustering coefficient C_{rand} is $10/1000 = 0.01$.

Randomly built graphs exhibit low clustering coefficients and are believed to represent something very close to the minimal possible average path length, at least in expectation. Perfectly ordered graphs, on the other hand, show high clustering coefficients but also high average path length. According to Watts and Strogatz, small world graphs lie between these two extremes: they exhibit high clustering coefficients, but short average path length, as shown in Equation (4.5).

$$\begin{aligned} &L \sim L_{rand} \\ &C \gg C_{rand} \ \ (C_{weight} \gg C_{rand}) \end{aligned} \tag{4.5}$$

This equation indicates the difference between a small world graph and a random graph: in a small world graph, there might exists "bundles" or highly interconnected groups. In this thesis, Equation (4.5) is called 'small world property'. If a graph fulfills these equations, it could be a small world graph.

4.1.2 Tuning Parameters Considering Small World Properties

Using different parameter combinations, we can build different structures of graphs. To optimize parameters, we have to consider the properties of small world graph for every set of parameter combination, then choose a graph that meets the small world properties best.

We modify the original requirement of small world property (Equation (4.5)) for optimization of parameters. When a graph has the property of small world, the average path length (L) should be almost equal to that of a random graph, while the clustering coefficient (C) should be great. These requirements can be represented as Equation (4.6) and (4.7).

$$\left| \frac{L}{L_{rand}} - 1 \right| \leq T_1 \tag{4.6}$$

$$C_{sw} = \left| \frac{C_{weight}}{C_{rand}} \right| \tag{4.7}$$

Equation (4.6) means that the difference between L and L_{rand} should be almost equal. T_1 is a threshold to control how we strictly check if L and L_{rand} are same. The range of T_1 is set to [0,1). While C_{sw} evaluates how C_{wight} is much greater than C_{rand}. For the small world graph, C_{sw} should be greater than 1 at least. It is preferable if C_{sw} is great. If $\left|\frac{L}{L_{rand}} - 1\right| \leq T_1$ and C_{sw} is very large, the structure of this graph satisfied the property of small world graph. On the other hand, if $\left|\frac{L}{L_{rand}} - 1\right|$ is great, or C_{sw} is less than 1, the structure of this graph does not satisfy the property of small world graph. The basic idea of the parameter optimization is to choose a graph that fulfills Equation (4.6) and C_{sw} is the highest for various combination of parameters. In other words, we search the graph that fulfills the small word property very well.

Although some graphs whose structures fulfill small world property, a large number of the hubs are detected. In other words, a large number of senses are induced. However, a word might not have an excessive number of senses. To choose the best parameter combination, we also consider to limit the number of hubs (senses) as in Equation (4.8)

$$Hubs \leq T_2 \tag{4.8}$$

, where $Hubs$ is the number of hubs, and T_2 is the other threshold for optimization of parameters.

Now the procedure to choose the best parameter combination considering small world property will be described. Firstly, we build the graphs for all parameter combinations. Parameter combination means a triplet of P_1, P_2 and P_3. Then, the parameter combinations are filtered out, if $\left|\frac{L}{L_{rand}} - 1\right|$ is infinite[1] or $\left|\frac{C_{weight}}{C_{rand}}\right|$ is less than 1. Next, choose the parameter combinations which fulfill that $\left|\frac{L}{L_{rand}} - 1\right|$ is not larger than T_1. Next,

[1]Note that L_{rand} is sometimes 0. In this case, $|\frac{L}{L_{rand}} - 1|$ becomes infinite.

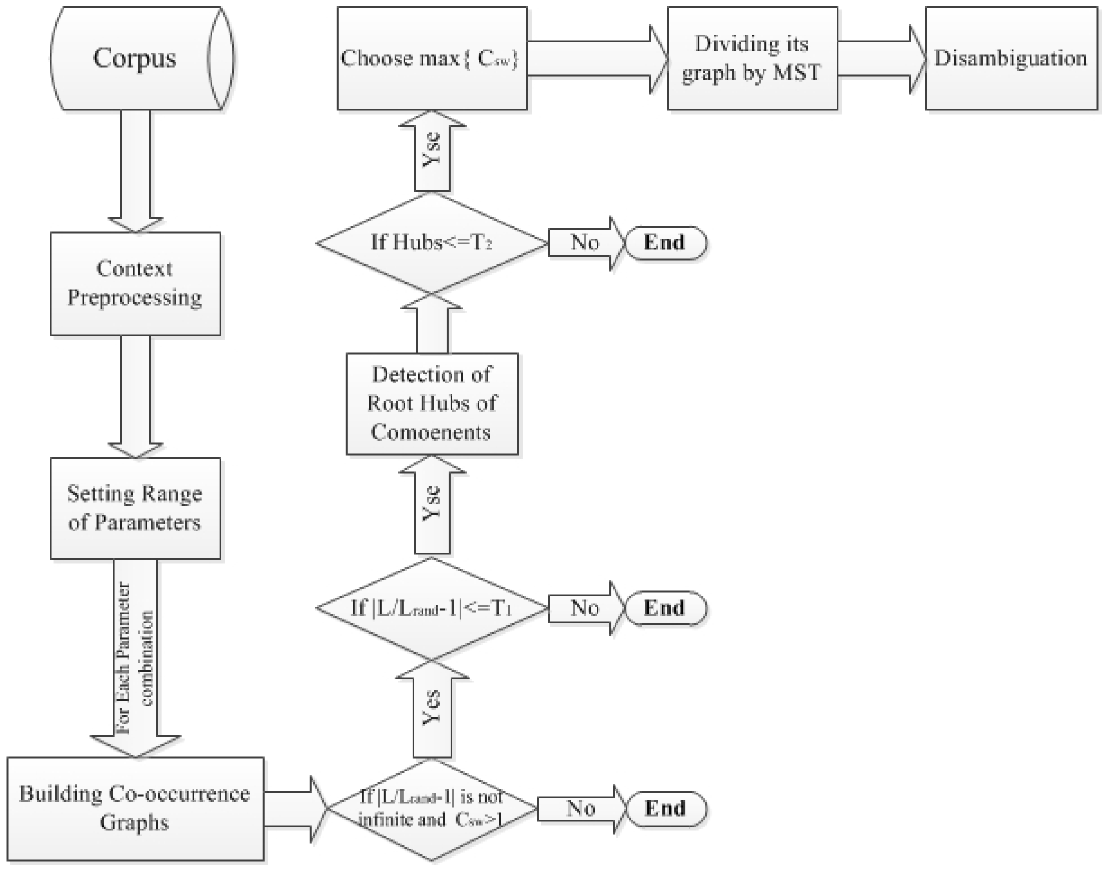
in order to guarantee the validity of clusters (senses), we will filter the parameter triplet where the number of the hubs is not greater than T_2. Finally, among remaining parameter combinations, we choose one where C_{sw} is maximum.

A pseudocode of the above algorithm is shown in Figure 4.1:

```
**Best Parameter**($PC$) {
    $PC$: all parameter combinations;
    $CPC$: candidate parameter combinations;
    $bpc$: the best parameter combination;
    $CPL(pc)$: function to return $|L/L_{rand} - 1|$ for $pc$;
    $CC(pc)$: function to return $|C/C_{rand}|$ for $pc$;
    $Hubs(pc)$: function to return number of hubs for $pc$;
    $CPC \leftarrow \varnothing$;
    $bpc \leftarrow empty$;
    for each $pc$ in $PC$ {
        if ($CPL(pc) < \infty$ and $CC(pc) > 1$ and $CPL(pc) <= T_1$ and $Hubs(pc) <= T_2$ ) {
            $CPC \leftarrow (CPC \cup pc)$; }};
    for each $pc$ in $CPC$ {
        if ($CC(pc) > CC(bpc)$) {
            $bpc \leftarrow pc$; }}
    Return $bpc$;
}
```

Figure 4.1: Algorithm of Parameter Optimization

In order to more clearly understand this proposed method, we show a flowchart of it in Figure 4.2.



Corpus
Context Preprocessing
Setting Range of Parameters
For Each Parameter combination
Building Co-occurrence Graphs
If |L/Lrand-1| is not infinite and Csw>1
No
End
Yes
If |L/Lrand-1|<=T1
No
End
Yse
Detection of Root Hubs of Comoenents
If Hubs<=T2
No
End
Yse
Choose max{ Csw}
Dividing its graph by MST
Disambiguation

Figure 4.2: Flowchart of Choosing the Best Parameter Combination

4.2 Finding Hubs

HyperLex finds a hub by simply considering the frequency of nodes. However, the connectivity between nodes in the graph is not considered. In order to analyze the connectivity in the co-occurrence graph and divide it into better sub-graphs, we used a graph connectivity measure - *Key Player Problem (KPP)* to delineate components. The key idea is that a node that are connected with many other nodes could be an appropriate hub.

4.2.1 Key Player Problem as Graph Connectivity Measure

In this subsection, we describe the graph connectivity measure that we apply for WSI.

Although the graph connectivity measures can be applied to both directed and undirected graphs, in the context of WSI, we assume that we are deal with undirected graphs. In an undirected graph G, two vertices u and v are called connected if G contains a path from u to v. Otherwise, they are called disconnected. If the two vertices are connected by a path of length 1, *i.e.* by a single edge, the vertices are called adjacent. A graph is said to be connected if every pair of vertices in the graph is connected. A connected component is a maximum connected subgraph of G. Each vertex belongs to exactly one connected component, as does each edge.

First, the distance between u and v is defined as:

$$d(u,v) = \begin{cases} length\ \ of\ \ the\ \ shortest\ \ path, & if\ u \to v \\ K, & otherwise \end{cases} \tag{4.9}$$

, where $u{\to}v$ indicates the existence of a path from u to v (u and v are connected), and K is a conventional constant, when u and v is disconnected. We choose $K{=}|V|$, as the length of any shortest path is smaller than V. The length of a path is calculated as the number of edges in the path. In an example in Figure 4.3, $d(a,i) = 5, d(c,g) = 4$, and so on.

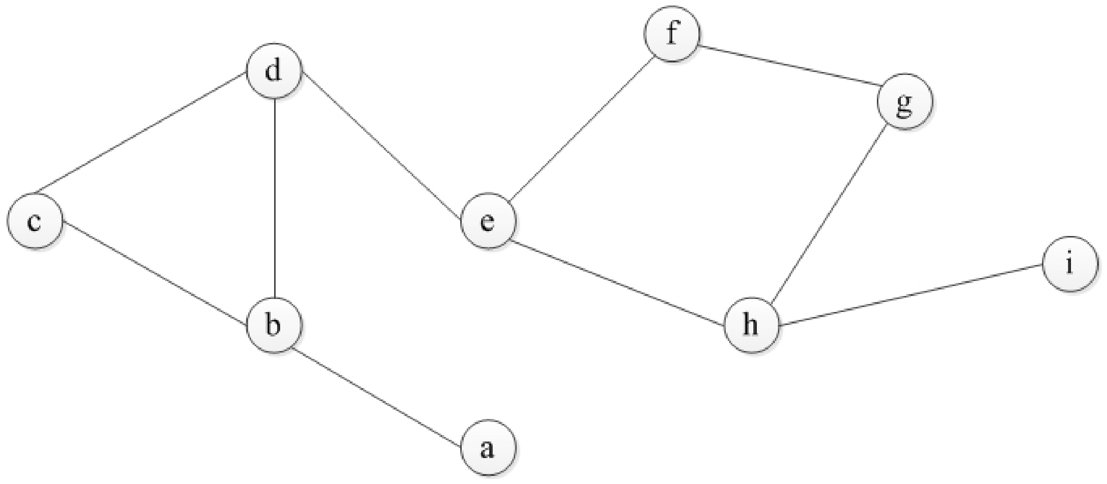


Figure 4.3: An Example of a Graph

Local measures of graph connectivity determine the degree of relevance of a single vertex v in a graph G. They can be viewed as measures of the influence of a node over the spread of information through the network [17]. Formally, we define a local measure l as:

$$l : V \rightarrow [0, 1] \tag{4.10}$$

A value close to one indicates that a vertex is important, whereas a value close to zero indicates that the vertex is peripheral.

Several local measures of graph connectivity have been proposed in the literature. A large number rely on the notion of centrality: a node is central if it is maximally connected to all other nodes. *Key Player Problem (KPP)* that is local graph connectivity measure will be considered to detect the components of the co-occurrence graph in this study.

The *key player problem (KPP)* consists of two separate sub-problems, which can be generally described as follows [18][2]:

- (*KPP-1*) Given a social network, find a set of k nodes (called a *kp-set* of order k) which, if removed, would maximally disrupt communication among the remaining nodes.
- (*KPP-2*) Given a social network, find a *kp-set* of order k that is maximally connected to all other nodes.

Of course, these introductory definitions leave out what is meant precisely by 'maximally disrupt communication' and 'maximally connected'. Part of the process of solving these problems is providing definitions of these concepts that lead to feasible solutions and useful outcomes. However, it would seem clear that *KPP-1* involves fragmenting a network into components, or, failing that, making distances between nodes so large as to be practically disconnected. In contrast, *KPP-2* involves finding nodes that can reach as many remaining nodes as possible via direct links or perhaps short paths.

The first problem, *KPP-1*, arises in a number of contexts. A prime example in the public health context is the immunization/quarantine problem. Given an infectious disease that is transmitted from person to person, and given that it is not feasible to immunize and/or quarantine an entire population, which subset of members should be immunized/quarantined so as to maximally hinder the spread of the infection? An example in the military context is target selection. Given a network of terrorists who must coordinate in order to mount effective attaches, and given that only a small number can be intervened with (*e.g.* by arresting or discrediting), which ones should be chosen in order to maximally disrupt the network?

The second problem, *KPP-2*, arises in the public health context when a health agency

[2]The following explanation of KPP is quoted from the paper [18].

needs to select a small set of population members to use as seeds for the diffusion of practices or attitudes that promote health, such as using bleach to clean needles. In the organizational management context, the problem occurs when management wants to implement a change initiative and needs to get a small set of informal leaders on-board first, perhaps by running a weekend intervention with them. In the military context, KPP-2 translates to locating an efficient set of enemies to surveil, turn (into double-agents), or feed misinformation to.

At first glance, both *KPP-1* and *KPP-2* would appear to be easily solved by either employing some graph theoretic concepts such as cutpoints and cutsets, or via the methods of social network analysis, such as measuring node centrality. It turns out, however, that none of the existing methods are adequate. The paper [18] explains why and presents a new approach specifically designed for the key player problem.

In a word, *KPP* is similar to the well known closeness centrality measure which is defined as the reciprocal of the total shortest distance from a given node to all other nodes. With *KPP*, as shown in Equation (4.11), a vertex v is considered important if it is relatively close to all other vertices [19]:

$$KPP\left(v\right) = \frac{\sum\limits_{u \in V : u \neq v} \frac{1}{d(u,v)}}{|V| - 1} \tag{4.11}$$

In (4.11), the numerator is the sum of the inverse shortest distances between v and all other nodes and the denominator is the number of nodes in the graph (excluding v). The *KPP* of a disconnected node is a small constant, given by $\frac{1}{K} = \frac{1}{V}$. For example, the *KPP* for vertices a and f in Figure 4.3 are $KPP\left(a\right) = \frac{1+\frac{1}{2}+\frac{1}{2}+\frac{1}{3}+\frac{1}{4}+\frac{1}{4}+\frac{1}{5}+\frac{1}{5}}{8-1} = 0.40$, $KPP\left(f\right) = \frac{1+1+\frac{1}{2}+\frac{1}{2}+\frac{1}{3}+\frac{1}{3}+\frac{1}{3}+\frac{1}{4}}{8-1} = 0.53$, respectively.

4.2.2 Finding the Hubs using *KPP*

The value of *KPP* for each vertex in the co-occurrence graph fully considers the connectivity in the graph, then the importance of vertices will be analyzed. If the valve of *KPP* for a vertex is very large, it indicates that this vertex is important in the graph. It will be considered as a candidate of a hub of a component.

Now the procedure to choose the hubs of the components using *KPP* will be described. First, we obtain the value of *KPP* for each vertex in the graph, then choose the vertex that have the largest *KPP* as the first hub. The chosen hub and its neighbors are removed from the graph. Next, we choose the vertex that have the greatest *KPP* in the remaining graph as the next hub and remove it and its neighbors from the graph. Note that *KPP* of nodes are calculated once and not re-evaluated during the iteration. We continue the above iteration process until the graph have no vertex.

We found that a node with the highest *KPP* has sometimes no neighboring node in the iterative procedure. Such hubs are obviously inappropriate since it is not a center of a dense sub-graph. The reason is that *KPP* values of nodes are calculated on the original graph. Even when *KPP* of a node is high, all neighbors of it can be removed if they are also neighbors of previously chosen hubs. Let us suppose that h_1, h_2 and h_3 have already chosen as the hubs and h_4 is selected as the next hub in an example graph in Figure 4.4. Note that the node ‘a’, ‘b’, ‘c’ and ‘d’ are connected to both h_3 and h_4. Since the hub and its neighboring nodes are removed from the graph in the iterative process, h_4 becomes isolated, even its *KPP* value is high in the original graph. h_4 is inappropriate as the hub, since it might belong to a component of the other hub. That is, h_3, h_4 and node ‘a’, ‘b’, ‘c’, ‘d’ seem to form one component. Therefore, we introduce a simple heuristics: in each iteration for finding hubs, we don’t consider a node that have no neighboring in the remaining graph as the hub, although its *KPP* value is high. Such a node is just removed and a node with the highest *KPP* among the rest of nodes is chosen as the next hub.

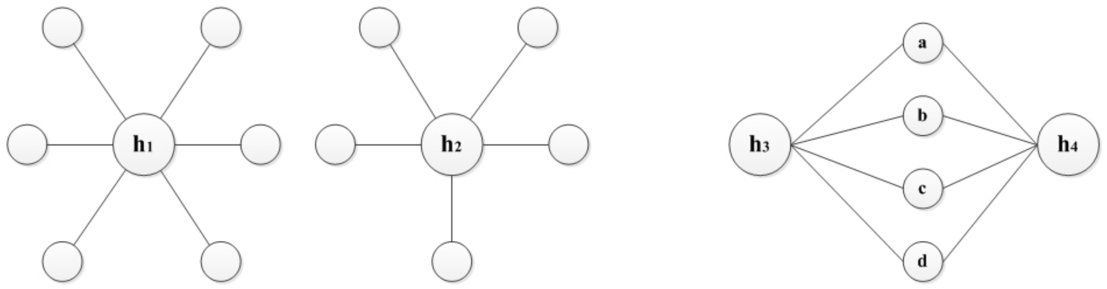


Figure 4.4: Example of Isolated Candidate of the Hub

A pseudocode of the procedure for detecting the hubs using *KPP* is shown in Figure 4.5:

```
Root Hubs($G$) {
    $G$: co-occurrence graph;
    $KPP$: array of $KPP$ of vertices in $G$;
    $V \leftarrow$ a set of nodes in $G$ sorted in descending order of $KPP$;
    $H \leftarrow \varnothing$;
    While ($V \neq \varnothing$) {
        $N \leftarrow$ the first node in $V$;
        If ($v$ have neighbors) {
            $H \leftarrow H \cup \{v\}$;
            $N \leftarrow$ the neighbors of $v$ and $v$ itself;
            $V \leftarrow V \setminus N$;
        } else {
            $V \leftarrow V \setminus \{v\}$;
        }
        return H;
    }
```

Figure 4.5: Algorithm for Detecting Hubs using *KPP*

4.3 Weighting Edges Considering Syntactic Relations

In HyperLex, a weight of an edge in a graph is simply determined based on the co-occurrence of two vertices (words), that is, the weight of the edge between two words only reflects the number of co-occurrence of two words in the corpus. However, even when two words appear in the same sentence, it is uncertain if they are strongly related each other. Co-occurrence of two words may happen by accident.

To solve this problem, syntactic relations are considered in the weighting scheme. If two words appear in a syntactic relation such as verb-(subject)-noun and verb-(object)-noun, they seem to more correlate. Therefore, low weight is given for the edge between two words when they appear under a syntactic relation many times in the corpus.

4.3.1 Syntactic Analysis with Standford Parser

To obtain the syntactic relation of two words on a sentence in the corpus including the target word, we parse the sentence using the Stanford CoreNLP that generates a parsing result in dependency relation format. Stanford CoreNLP provides a set of natural language analysis tools including *Part-of Speech (POS) Tagger*, *Named Entity Recognizer (NER)*, *Stanford Parser, Coreference Resolution System*, *Sentiment Analysis*, and *Bootstrapped Pattern Learning* tools as well as model files for analysis of English. In this research, the Stanford Parser was used to analyze grammatical relations for the instances in the corpus. Stanford Parser provides 55 different grammatical dependency types as shown in Table 4.1.

Label	Grammatical Relations	Label	Grammatical Relations
abbrev	abbreviation modifier	**npadvmod**	noun phrase as adverbial modifier
acomp	adjectival complement	**nsubj**	nominal subject
advcl	adverbial clause modifier	**nsubjpass**	passive nominal subject
advmod	adverbial modifier	**num**	numeric modifier
agent	agent	**number**	element of compound number
amod	adjectival modifier	**parataxis**	parataxis: parataxis
appos	appositional modifier	**partmod**	participial modifier
attr	attributive	**pcomp**	prepositional complement
aux	auxiliary	**pobj**	object of a preposition
auxpass	passive auxiliary	**poss**	possession modifier
cc	coordination	**possessive**	possessive modifier
ccomp	clausal complement	**preconj**	preconjunct
complm	complementizer	**predet**	predeterminer
conj	conjunct	**prep**	prepositional modifier
cop	copula	**prepc**	prepositional clausal modifier
csubj	clausal subject	**prt**	phrasal verb particle
csubjpass	clausal passive subject	**punct**	punctuation
dep	dependent	**purpcl**	purpose clause modifier
det	determiner	**quantmod**	quantifier phrase modifier
dobj	direct object	**rcmod**	relative clause modifier
expl	expletive	**ref**	referent
infmod	infinitival modifier	**rel**	relative
iobj	indirect object	**root**	root
mark	marker	**tmod**	temporal modifier
mwe	multi-word expression	**xcomp**	open clausal complement
neg	negation modifier	**xsubj**	controlling subject
nn	noun compound modifier		

Table 4.1: Labels of Grammatical Relations in Stanford Parser

The parser can read various forms of plain text as input and can output various analysis formats, including part-of-speech tagged text, phrase structured trees, and a grammatical relations (typed dependency) format. Table 4.2 shows an example sentence and its POS-tagged text.

Text
The strongest rain ever recorded in India shut down the financial hub of Mumbai, snapped communication lines, closed airports and forced thousands of people to sleep in their offices or walk home during the night, officials said today.
Part-of-Speech Tagged Text
The/DT strongest/JJS rain/NN ever/RB recorded/VBN in/IN India/NNP shut/VBD down/RP the/DT financial/JJ hub/NN of/IN Mumbai/NNP ,/, snapped/VBD communication/NN lines/NNS ,/, closed/VBD airports/NNS and/CC forced/VBD thousands/NNS of/IN people/NNS to/TO sleep/VB in-/IN their/PRP$ offices/NNS or/CC walk/VB home/NN during/IN the/DT night/NN ,/, officials/NNS said/VBD today/NN ./.

Table 4.2: Example Sentence and its Results of POS Tagging

If two words in a sentence appear in any grammatical relations, the parser will output a table of typed dependencies. Table 4.3 shows the list of typed dependencies obtained for the example sentence in Table 4.2. Grammatical relations are represented in a form of $relation(word_1, word_2)$, showing that the relation between $word_1$ and $word_2$ is '$relation$'. Note that an index (a position in a sentence) of each word is also shown by Standford Parser.

typed dependencies	typed dependencies
det(rain-3, The-1)	cc(shut-8, and-22)
amod(rain-3, strongest-2)	conj(shut-8, forced-23)
nsubj(shut-8, rain-3)	dobj(forced-23, thousands-24)
advmod(recorded-5, ever-4)	prep(thousands-24, of-25)
vmod(rain-3, recorded-5)	pobj(of-25, people-26)
prep(recorded-5, in-6)	aux(sleep-28, to-27)
pobj(in-6, India-7)	xcomp(forced-23, sleep-28)
ccomp(said-40, shut-8)	prep(sleep-28, in-29)
prt(shut-8, down-9)	poss(offices-31, their-30)
det(hub-12, the-10)	pobj(in-29, offices-31)
amod(hub-12, financial-11)	cc(sleep-28, or-32)
dobj(shut-8, hub-12)	conj(sleep-28, walk-33)
prep(hub-12, of-13)	dobj(walk-33, home-34)
pobj(of-13, Mumbai-14)	prep(walk-33, during-35)
conj(shut-8, snapped-16)	det(night-37, the-36)
nn(lines-18, communication-17)	pobj(during-35, night-37)
dobj(snapped-16, lines-18)	nsubj(said-40, officials-39)
conj(shut-8, closed-20)	root(ROOT-0, said-40)
dobj(closed-20, airports-21)	tmod(said-40, today-41)

Table 4.3: List of Grammatical Dependencies for the Example Text

4.3.2 Weighting Scheme

In HyperLex, the weight of the edge in the co-occurrence graph represents the correlation between two words. Although the weights are determined according to the co-occurrence of two words in HyperLex, our weighting scheme considers both co-occurrence and syntactic relation.

In order to more clearly understand the new weighting scheme under syntactic relation, we consider an example sentence "*we drink water and eat meat at the party.*" Figure 4.6 shows the basic dependencies of this sentence. We observe two pairs of words in this sentence, 'drink-water' and 'drink-meat'. The weights of their edges may not be so different in using the algorithm of HyperLex, however, 'drink-meat' do not appear in any syntactic relation, and they seem less correlated than 'drink-water'.

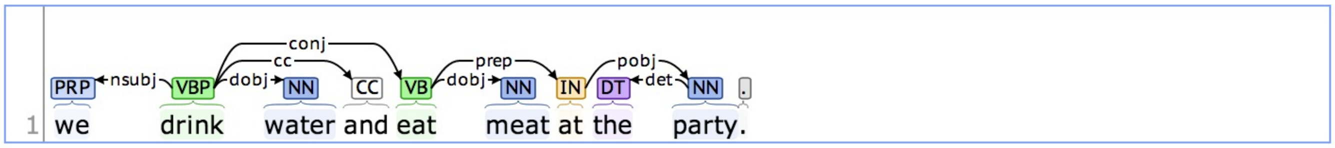


Figure 4.6: Basic Dependencies of a Sentence

To consider syntactic relations in the weighting scheme, the weight assigned to each edge is redefined as Equation (4.12).

$$w_{ij}^{syn} = w_{ij} \cdot \left(1 - \frac{occur_{syn}\left(n_i, n_j\right)}{occur\left(n_i, n_j\right)}\right) \tag{4.12}$$

w_{ij}^{syn} stands for the new weight, while, w_{ij} represents the original weight in the HyperLex (Equation (3.1)). $occur_{syn}\left(n_i, n_j\right)$ represents the number of sentences where n_i and n_j are under some syntactic relations, while $occur(n_i, n_j)$ is the number of sentences where n_i and n_j cooccur. Note that $occur_{syn}(n_i, n_j)$ is the sum of 55 grammatical relations.

Equation (4.12) means that the weight of two words becomes small if two words n_i and n_j more appear in the syntactic relations. The new weighting scheme enables us to make components (or sub-graphs) consisting of words that are syntactically related each other. Such components may more precisely represent the sense of the target word or the set of words in the context of the sense.

Chapter 5

Evaluation

This chapter reports the evaluation of the proposed method. First, the data used for the evaluation is introduced. Then, measures to evaluate the performance of WSI are described. Next, the setting of the experiment is explained. Finally, the results of WSI by the original and revised HyperLex are compared to show the effectiveness of the proposed method.

5.1 Data

In this research, the training data of Senseval-3 English Lexical Sample Task is used for the evaluation. Senseval-3 is an evaluation workshop for Word Sense Disambiguation. The data is a collection of documents where a correct sense of a target word is annotated. In other words, it is a sense tagged corpus. The number of the target word is 57. The list of the target words and the number of samples or instances for each target word is shown in Table 5.1.

Target Word	**No. of Instances**	**Target Word**	**No. of Instances**
activate	228	miss	58
add	263	note	132
appear	265	operate	35
argument	221	organization	112
arm	266	paper	232
ask	261	party	230
atmosphere	161	performance	172
audience	200	plan	166
bank	262	play	104
begin	181	produce	186
climb	133	provide	136
decide	122	receive	52
degree	256	remain	139
difference	226	rule	59
different	98	shelter	196
difficulty	46	simple	36
disc	200	solid	108
eat	181	smell	58
encounter	130	sort	190
expect	156	source	64
express	110	suspend	128
hear	63	talk	146
hot	86	treat	112
image	146	use	26
important	36	wash	66
interest	185	watch	100
judgment	62	win	78
lose	71	write	44
mean	80		

Table 5.1: Number of Instances for Each Target Word

In this experiment, instances for each target word are used as a corpus to infer the senses of the target word. As shown in Table 5.1, the sizes of the corpora are different for the target words. It is appropriate for the evaluation of the proposed method, since the co-occurrence graph depends on the size of the corpus and our method focuses on how to optimize parameters to construct a better graph. Figure 5.1 shows the distribution of the size of the corpus. It indicates that the proportions of the small, medium and large corpora are roughly equal.

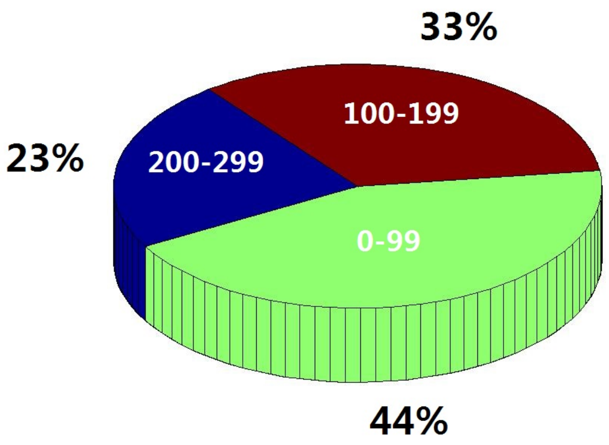


Figure 5.1: Distribution of the Size of Corpora

Context Preprocessing

Before applying HyperLex, two preprocessings are performed on the text: lemmatization and removal of stop word.

Lemmatisation in linguistics is the process of grouping together the different inflected forms of a word so they can be analyzed as a single item. In computational linguistics, lemmatisation is the algorithmic process of determining the lemma for a given word. Since the process may involve complex tasks such as understanding context and determining parts of speech of words in a sentence. It is the process for reducing inflected (or sometimes derived) words to their stems, base or root forms - generally a written word form. In our research, we use the tool "*European Languages Lemmatizer*" for this task. It finds all paradigms and their forms of a word and give morphological information - part of speech, case, gender, tense, etc. The first step of lemmatization is to create a big dictionary that maps words to their stems. The second step is to use a set of rules that extract stems from words.

Stop word is a word that do not have a meaning and should be ignored in the most of natural language processing. There are 3 kinds of stop words. The first is the verbs that have just a general meaning such as 'be', 'do' and so on. The second is function words (determiners, preposition, etc.). The last is some meaningless words in Senseval-3 corpus (such as '85dbs', 'pp.', 'Fig.' and so on). These non-informative words that are general meaning verbs and function words are defined by a "stopword list"[1]. For each word in

[1]https://sites.google.com/site/kevinbouge/stopwords-lists

the corpus, we search it in the stopword list one by one. When the word is found in the list, it is removed. This process takes time $O(n)$. For the meaningless words in Senseval-3 corpus, if the words are the combinations of numbers and letters (e.g.'85dbs') or the combinations of letters and symbols (e.g. 'Fig.') or not larger than two letters (e.g. 'pp'), we will filter out them.

5.2 Evaluation Measure

Word Sense Induction can be regarded as a kind of clustering. It is a task to construct several clusters of instances of the target word where each cluster consists of instances of the same meaning. In this experiment, the performance of WSI is measured from a point of view of clustering. The gold standard is a set of clusters consisting of the instances where the annotated sense in the corpus are same. While WSI method (HyperLex or our proposed method) also produces clusters. If the generated clusters are same or similar to the gold clusters, the WSI method can be regarded that it could infer the senses of the target word. The most popular measures for clustering evaluation are Purity and Inverse Purity. Purity focuses on the frequency of the most common category in each cluster, and rewards the clustering solutions that introduce less noise in each cluster. Being C the set of clusters to be evaluated, L the set of categories (manually annotated) and N the number of clustered elements, purity [20] is computed by taking the weighted average of maximal precision values as shown in Equation (5.1):

$$Purity = \sum_i \frac{|C_i|}{N} \max_i Precision\,(C_i, L_j) \tag{5.1}$$

$Precision(C_i, l_j)$, the precision of a cluster C_i for a given category L_j, is defined as Equation (5.2).

$$Precision\,(C_i, L_j) = \frac{|C_i \cap L_j|}{|C_i|} \tag{5.2}$$

Purity penalizes the noise in a cluster, but it does not reward grouping items from the same category together. If we simply make one cluster per item, we reach trivially a maximum purity value. Inverse Purity focuses on the cluster with maximum precision for each category. Inverse Purity [20] is defined as Equation (5.3).

$$Inverse\,Purity = \sum_i \frac{|L_i|}{N} \max_i Precision\,(L_i, C_j) \tag{5.3}$$

Inverse Purity rewards grouping items together, but it does not penalize mixing items from different categories. We can reach a maximum value for Inverse Purity by making a single cluster with all items.

Purity and Inverse Purity is under trade-off. When Purity is high, Inverse Purity tends to be low, and vice verse. To consider both Purity and Inverse Purity, we use the harmonic mean of them, *P-IP* [21], as shown in (5.4). Equation (5.4) is similar to *F-measure*, which is defined as the harmonic mean of precision and recall, used for evaluation of Information Retrieval (IR).

$$P\text{-}IP = \frac{2 \times Purity \times InversePurity}{Purity + InversePurity} \tag{5.4}$$

5.3 Setting of Experiment

5.3.1 Range of Parameters

In our WSI method, we optimize three parameters to build the co-occurrence graph:

- A threshold of a length of a sentence to be used to make a graph (parameter P_1)
- A threshold of a frequency of a word to be added as a node (parameter P_2)
- A threshold of a co-occurrence frequency of a pair of words to be added as an edge (parameter P_3)

Table 5.2 shows ranges of the parameters considered in our optimization procedure. Note that three parameters are integers. A value of each parameter is changed in the range shown in Table 5.2. Thus we tried $14 \times 15 \times 15 = 3570$ combinations of parameters, and chose the best combination of parameters by considering the conditions of small world properties.

Parameter	**Range**
P_1	[3 , 16]
P_2	[4 , 20]
P_3	[2 , 16]

Table 5.2: Range of Parameters

5.3.2 Threshold

In our method that optimizes the parameters, we need to set two thresholds T_1 and T_2. Recall that T_1 and T_2 are the upper bound of the index $\left|\frac{L}{L_{rand}} - 1\right|$ and the upper bound of the number of the hubs, respectively. Table 5.3 shows how to set them in the experiment.

Threshold	**Range**
T_1	$AVG(\left|\frac{L}{L_{rand}} - 1\right|)$
T_2	5, 6, 7, 8, 9, 10, 15, 20, 25, 30, $AVG(Hubs)$

Table 5.3: Range of Thresholds

$AVG(\left|\frac{L}{L_{rand}} - 1\right|)$ represents the average of $\left|\frac{L}{L_{rand}} - 1\right|$ of the graphs for all combination of parameters. While $AVG(Hubs)$ represents the average number of the hubs of all the graph that the system can find components for various parameter combination. We will evaluate our WSI methods for $T_1 = AVG(\left|\frac{L}{L_{rand}} - 1\right|)$ and various values of T_2.

5.4 Result of Baseline

In this research, we considered the result of the original algorithm of HyperLex as the baseline. Parameters are set as $p_1 = 4$, $p_2 = 10$ and $p_3 = 5$, which are default parameters in HyperLex. Clusters of the instances are made for 57 target words. Then Purity, Inverse Purity and $P\text{-}IP$ are measured. Table 5.4 and 5.5 show the results of HyperLex for each target word.

Target Word	N	E	$\frac{C}{C_{rand}}$	$\frac{L}{L_{rand}} - 1$	Hub	P	IP	P-IP
activate	109	284	12.506	0.458	41	0.807	0.421	0.553
add	114	201	22.412	0.673	46	0.616	0.437	0.511
appear	138	445	10.621	0.410	58	0.649	0.475	0.549
argument	146	345	18.178	0.505	63	0.611	0.362	0.455
arm	150	678	7.544	0.235	48	0.797	0.241	0.370
ask	69	145	9.540	0.567	28	0.487	0.579	0.529
atmosphere	52	108	6.994	0.506	19	0.540	0.280	0.368
audience	135	509	8.091	0.274	46	0.820	0.290	0.428
bank	203	958	9.152	0.281	91	0.813	0.302	0.440
begin	73	219	6.377	0.387	30	0.580	0.354	0.439
climb	29	35	7.195	0.826	10	0.617	0.286	0.390
decide	18	19	6.672	0.949	7	0.811	0.320	0.459
degree	181	576	15.507	0.420	69	0.734	0.422	0.536
difference	121	319	12.514	0.471	30	0.482	0.757	0.589
different	18	21	4.863	0.878	7	0.541	0.316	0.399
difficulty	9	10	2.878	0.917	3	0.522	0.543	0.532
disc	129	416	9.308	0.446	50	0.680	0.435	0.531
eat	53	112	7.428	0.570	21	0.856	0.481	0.616
encounter	37	62	6.514	0.657	11	0.631	0.431	0.512
expect	37	50	8.892	0.782	23	0.756	0.282	0.411
express	39	43	12.929	0.903	19	0.673	0.245	0.360
hear	2	1	1.333	2.000	1	0.762	0.587	0.663
hot	4	2	2.417	1.500	2	0.907	0.360	0.516
image	79	244	5.846	0.241	25	0.658	0.390	0.490
important	4	3	1.588	1.311	1	0.611	0.778	0.684
interest	102	188	16.678	0.630	46	0.486	0.351	0.408
judgment	11	18	2.036	0.642	3	0.484	0.806	0.605
lose	2	1	1.231	2.000	1	0.831	0.324	0.466

Table 5.4: Results of *HyperLex* for All Target Words (1)

Target Word	N	E	$\left\|\frac{C}{C_{rand}}\right\|$	$\left\|\frac{L}{L_{rand}} - 1\right\|$	Hub	P	IP	$P\text{-}IP$
mean	18	20	5.216	0.895	7	0.650	0.438	0.523
miss	10	13	2.524	0.762	5	0.603	0.362	0.453
note	36	62	6.694	0.639	9	0.523	0.667	0.586
operate	0	0	NaN	NaN	0	1.000	0.086	0.158
organization	59	114	8.774	0.512	21	0.857	0.223	0.354
paper	118	298	13.166	0.486	51	0.444	0.448	0.446
party	169	983	4.843	0.197	45	0.757	0.774	0.765
performance	76	189	8.210	0.492	31	0.465	0.372	0.413
plan	91	223	10.232	0.486	33	0.771	0.422	0.545
play	19	19	6.143	1.000	12	0.615	0.298	0.402
produce	83	146	16.550	0.654	37	0.677	0.306	0.422
provide	57	166	4.534	0.364	21	0.860	0.316	0.462
receive	16	16	5.172	1.000	5	0.846	0.558	0.672
remain	53	89	9.516	0.581	28	0.871	0.173	0.288
rule	16	21	3.467	0.781	8	0.814	0.576	0.675
shelter	84	167	12.425	0.528	37	0.582	0.148	0.236
simple	0	0	NaN	NaN	0	1.000	0.139	0.244
solid	0	0	NaN	NaN	0	1.000	0.207	0.343
smell	22	23	6.755	0.961	7	0.611	0.407	0.489
sort	75	185	10.217	0.494	31	0.642	0.479	0.549
source	7	5	3.850	1.173	3	0.734	0.375	0.496
suspend	51	83	10.837	0.699	14	0.445	0.766	0.563
talk	33	59	5.999	0.625	17	0.815	0.363	0.502
treat	21	19	7.591	1.093	11	0.580	0.393	0.469
use	0	0	NaN	NaN	0	1.000	0.192	0.323
wash	3	2	2.059	1.492	1	0.758	0.515	0.613
watch	13	14	3.564	0.935	5	0.790	0.410	0.540
win	5	4	2.847	1.139	2	0.718	0.372	0.490
write	11	16	2.726	0.685	2	0.614	0.727	0.666

Table 5.5: Results of *HyperLex* for All Target Words (2)

In Table 5.4 and Table 5.5, N and E respectively represent the number of nodes and edges in the graph, and Hub represents the number of hubs or induced senses. P, IP and $P\text{-}IP$ are Purity, Inverse Purity and the harmonic mean of them, respectively. We found that we cannot detect any hubs for some target words (*e.g.* the target word 'operate', 'use') or some graphs did not fulfill the properties of small world graph for some target words (namely, $\left|\frac{L}{L_{rand}} - 1\right| \geq 1$, *e.g.* the target word 'hear', 'hot'). The former means that a graph can not be constructed since no word is added as a node, while the latter means that a graph is sparse. The reason is that HyperLex uses a fixed parameters to perform WSI, but they are not adapted for the size of the corpus for some target words.

5.5 Results of the Proposed Method

5.5.1 Results of Optimization of Parameters

The effectiveness of the optimization of parameters described in Section 4.1 is examined. We call HyperLex using optimized parameters *HyperLex_OP*. Table 5.6 and 5.7 reveal that results of *HyperLex_OP* for all target words. In this experiment, the threshold T_2 is set to AVG(Hub).

Target Word	P_1	P_2	P_3	N	E	$\frac{C}{C_{rand}}$	$\frac{L}{L_{rand}} - 1$	Hub	P	IP	P-IP
activate	3	13	5	96	265	10.41	0.43	34	0.803	0.465	0.589
add	14	15	5	67	139	12.33	0.58	23	0.563	0.635	0.597
appear	3	4	7	61	118	10.24	0.57	26	0.657	0.257	0.369
argument	15	14	5	97	272	10.39	0.41	38	0.561	0.421	0.481
arm	16	12	7	73	216	5.87	0.33	26	0.793	0.267	0.399
ask	10	16	6	38	68	7.85	0.61	15	0.487	0.533	0.509
atmosphere	3	12	5	44	96	5.69	0.47	15	0.534	0.329	0.407
audience	3	12	7	62	134	7.82	0.49	25	0.755	0.315	0.445
bank	3	14	8	81	174	11.09	0.55	34	0.679	0.313	0.429
begin	3	9	6	45	116	5.26	0.44	14	0.575	0.337	0.425
climb	12	12	4	31	58	5.42	0.53	10	0.617	0.353	0.449
decide	15	9	4	38	73	6.18	0.53	15	0.779	0.295	0.428
degree	3	15	7	65	126	11.59	0.57	21	0.711	0.383	0.498
difference	3	12	6	80	156	11.87	0.55	20	0.496	0.681	0.574
different	12	10	4	27	54	3.93	0.51	7	0.541	0.316	0.399
difficulty	3	7	3	27	79	2.02	0.31	6	0.543	0.609	0.574
disc	3	14	8	49	90	7.84	0.64	18	0.660	0.425	0.517
eat	13	12	5	43	100	5.50	0.51	13	0.862	0.459	0.599

Table 5.6: Results of *HyperLex_OP* for All Target Words (1)

Target Word	P_1	P_2	P_3	N	E	$\frac{C}{C_{rand}}$	$\frac{L}{L_{rand}} - 1$	Hub	P	IP	P-IP
encounter	3	12	5	30	54	5.15	0.60	7	0.615	0.462	0.527
expect	11	10	4	59	129	9.10	0.49	20	0.750	0.391	0.514
express	3	10	4	55	113	8.62	0.51	21	0.664	0.227	0.339
hear	3	7	3	18	34	3.09	0.54	4	0.524	0.730	0.610
hot	3	10	3	14	23	2.91	0.48	5	0.826	0.349	0.490
image	3	12	6	44	96	6.06	0.49	18	0.616	0.356	0.451
important	3	6	3	28	66	2.88	0.50	3	0.444	0.944	0.604
interest	3	13	5	63	140	9.05	0.51	23	0.449	0.378	0.411
judgment	3	9	4	16	32	2.76	0.55	3	0.484	0.855	0.618
lose	3	7	3	44	126	3.25	0.31	11	0.521	0.338	0.410
mean	11	11	4	26	49	3.73	0.47	7	0.650	0.513	0.573
miss	3	8	4	14	23	2.52	0.55	6	0.552	0.397	0.461
note	3	11	4	56	145	6.56	0.47	18	0.591	0.636	0.613
operate	3	6	2	18	42	1.85	0.34	5	0.800	0.571	0.667
organization	3	10	4	87	323	4.60	0.25	26	0.813	0.339	0.479
paper	3	13	6	63	145	8.57	0.51	23	0.401	0.565	0.469
party	3	9	8	80	248	5.09	0.44	13	0.713	0.809	0.758
performance	3	13	6	49	84	9.86	0.62	23	0.453	0.273	0.341
plan	15	13	5	61	170	6.09	0.40	19	0.747	0.416	0.534
play	14	7	4	53	113	5.12	0.39	15	0.538	0.327	0.407
produce	3	12	4	87	289	7.47	0.35	25	0.640	0.452	0.529
provide	3	13	5	39	134	2.68	0.33	10	0.831	0.485	0.613
receive	3	7	4	32	62	4.48	0.49	6	0.846	0.500	0.629
remain	3	12	5	41	73	7.47	0.49	22	0.863	0.144	0.247
rule	3	11	4	18	34	3.07	0.56	6	0.797	0.593	0.680
shelter	16	11	5	76	153	11.50	0.51	33	0.597	0.133	0.217
simple	3	7	2	10	18	1.96	0.54	3	0.694	0.556	0.617
solid	3	9	2	12	23	2.41	0.52	2	0.448	0.672	0.538
smell	3	13	4	19	31	4.75	0.62	5	0.574	0.417	0.483
sort	15	7	6	48	99	6.51	0.56	15	0.637	0.500	0.560
source	3	10	3	16	30	2.68	0.49	5	0.641	0.375	0.473
suspend	3	10	4	65	177	6.05	0.44	13	0.445	0.734	0.554
talk	3	11	4	39	98	4.60	0.46	14	0.808	0.486	0.607
treat	3	9	3	49	149	4.35	0.30	14	0.554	0.554	0.554
use	3	6	2	6	10	1.18	0.62	1	0.731	0.692	0.711
wash	12	8	3	24	45	4.16	0.52	8	0.591	0.500	0.542
watch	3	12	3	17	40	2.46	0.44	4	0.760	0.630	0.689
win	16	8	3	39	97	4.08	0.36	13	0.577	0.308	0.401
write	3	9	4	15	32	1.63	0.48	3	0.614	0.705	0.656

Table 5.7: Results of *HyperLex_OP* for All Target Words (2)

In Table 5.6 and 5.7, P_1, P_2, P_3 are the best parameters for each target word. They are quite different for target words. It indicates that the parameter optimization is significant in HyperLex.

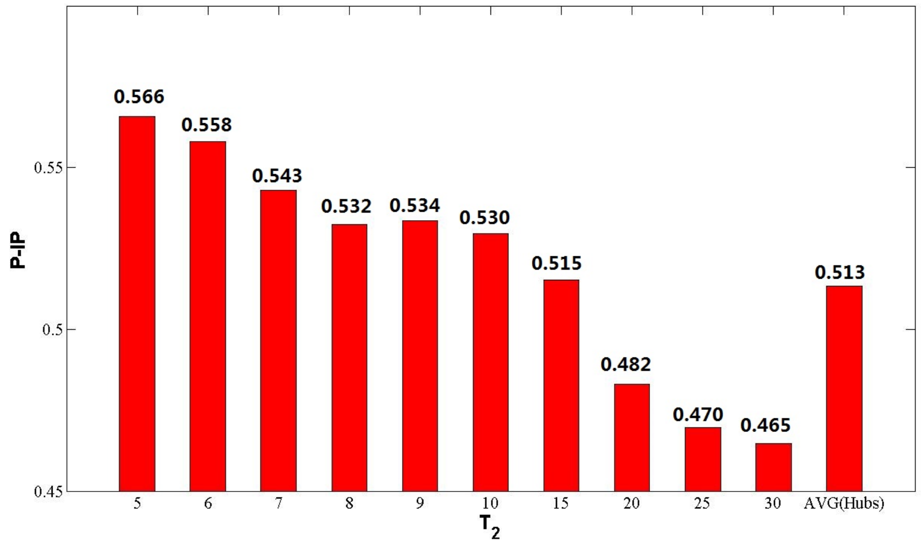


Figure 5.2: Results of *HyperLex_OP* for Different T_2.

Figure 5.2 compares the performance of *HyperLex_OP* for different threshold T_2. The horizontal axis indicates T_2, while the vertical axis indicates the average of $P\text{-}IP$ for all target words. The lower T_2 is set, the higher $P\text{-}IP$ is. When T_2 is set to be low, however, no graph fulfills the condition $Hubs \leq T_2$ (Equation (4.7)) for several target words. That is, *HyperLex_OP* fails to infer the senses. Table 5.8 shows the values of T_2 and the number of target words where no graph is obtained.

T_2	5	6	7	8	9	10	15	20	25	30	$AVG(Hubs)$
$No\ Graph$	15	11	7	4	2	1	1	0	0	0	0

Table 5.8: Number of Target Words for which No Graph is Obtained

Obviously, such cases are meaningless. When T_2 is set to 20, 25, 30 and AVG(Hubs), *HyperLex_OP* can infer the sense for all 57 target words. Among them, $T_2 = AVG(Hubs)$ achieved the best $P\text{-}IP$.

WSI Method	Hub	P	IP	P-IP
Baseline	21.8	0.698	0.410	0.482
HyperLex_OP	14.6	0.639	0.469	0.513

Table 5.9: Comparison of Baseline and *HyperLex_OP*

Table 5.9 shows the comparison of baseline and *HyperLex_OP* ($T_2 = AVG(Hubs)$). The values of Hub, P, IP, P-IP are the average result of 57 target words. Optimization of parameters improves Inverse Purity, but decreases Purity. However, *HyperLex_OP* outperforms the baseline in terms of P-IP. We can conclude that the parameter optimization is effective. In addition, the number of Hubs in *HyperLex_OP* is less than the Baseline. That is, *HyperLex_OP* infers less number of senses.

5.5.2 Results of Detection of Components Using KPP

In this subsection, our proposed method to choose the hub by *KPP* is evaluated. *HyperLex_OP+KPP* stands for the WSI method using *KPP* with parameter optimization. Table 5.10 and 5.11 show the results of *HyperLex_OP+KPP* for individual target words. Note that T_2 is set to $AVG(Hubs)$.

Target Word	P_1	P_2	P_3	N	E	$\frac{C}{C_{rand}}$	$\frac{L}{L_{rand}} - 1$	Hub	P	IP	P-IP
activate	3	18	6	46	93	8.768	0.535	5	0.811	0.715	0.760
add	3	14	6	42	72	9.055	0.667	2	0.548	0.939	0.692
appear	3	9	6	92	215	11.836	0.506	5	0.592	0.581	0.587
argument	15	12	5	121	309	14.261	0.461	10	0.493	0.683	0.573
arm	3	12	6	107	365	6.974	0.309	8	0.774	0.500	0.608
ask	16	11	5	58	116	8.773	0.580	5	0.533	0.510	0.521
atmosphere	16	10	4	79	247	6.574	0.356	5	0.516	0.696	0.592
audience	3	4	7	67	139	8.504	0.522	5	0.740	0.685	0.711
bank	3	14	8	81	174	11.091	0.547	5	0.653	0.798	0.718
begin	3	14	7	29	60	4.426	0.534	2	0.514	0.702	0.593
climb	12	12	4	31	58	5.420	0.534	6	0.549	0.346	0.424
decide	15	9	4	38	73	6.177	0.527	4	0.779	0.541	0.638
degree	3	11	6	114	274	14.071	0.490	7	0.641	0.594	0.616
difference	3	7	5	134	336	14.002	0.502	7	0.465	0.854	0.602
different	3	7	4	43	75	7.352	0.616	2	0.520	0.939	0.670
difficulty	3	7	3	27	79	2.019	0.313	3	0.413	0.913	0.569
disc	3	4	7	77	149	10.106	0.651	2	0.410	0.965	0.575
eat	3	9	5	54	113	7.580	0.577	3	0.845	0.934	0.887
encounter	3	11	4	49	133	5.239	0.413	4	0.554	0.554	0.554
expect	11	11	4	45	111	5.947	0.457	5	0.744	0.801	0.771
express	3	12	4	42	94	5.698	0.457	8	0.655	0.364	0.468
hear	3	9	2	14	35	1.161	0.461	2	0.508	0.905	0.651
hot	3	10	3	14	23	2.907	0.477	3	0.826	0.523	0.641
image	3	7	5	105	304	7.550	0.279	5	0.589	0.603	0.596
important	3	6	3	28	66	2.882	0.501	1	0.417	0.972	0.583
interest	3	15	6	36	64	6.926	0.598	4	0.395	0.670	0.497
judgment	3	7	4	25	42	4.291	0.662	1	0.419	0.903	0.573
lose	3	8	3	34	90	2.789	0.334	4	0.507	0.577	0.540

Table 5.10: Results of *HyperLex_OP+KPP* for All Target Words (1)

Target Word	P_1	P_2	P_3	N	E	$\frac{C}{C_{rand}}$	$\frac{L}{L_{rand}} - 1$	Hub	P	IP	$P\text{-}IP$
mean	3	7	4	35	63	5.180	0.524	3	0.613	0.650	0.631
miss	3	8	4	14	23	2.518	0.549	2	0.483	0.638	0.550
note	3	9	4	75	170	10.236	0.538	4	0.523	0.932	0.670
operate	3	5	2	29	81	1.995	0.212	3	0.629	0.600	0.614
organization	3	8	4	116	379	7.643	0.311	9	0.786	0.330	0.465
paper	3	6	5	136	323	15.280	0.515	5	0.315	0.841	0.458
party	16	10	7	102	379	5.299	0.340	3	0.704	0.865	0.777
performance	3	9	5	81	199	8.671	0.497	6	0.331	0.913	0.486
plan	3	11	5	80	209	8.223	0.445	6	0.735	0.428	0.541
play	14	7	4	53	113	5.119	0.394	6	0.490	0.433	0.460
produce	3	15	5	44	94	7.314	0.519	6	0.651	0.747	0.696
provide	3	10	6	39	87	4.068	0.480	3	0.824	0.728	0.773
receive	3	9	4	25	49	3.715	0.457	3	0.846	0.596	0.699
remain	3	12	5	41	73	7.470	0.486	5	0.856	0.338	0.485
rule	3	12	4	16	30	2.650	0.556	2	0.797	0.797	0.797
shelter	14	9	5	87	168	13.157	0.547	11	0.531	0.265	0.354
simple	3	7	2	10	18	1.955	0.541	2	0.722	0.583	0.645
solid	3	9	2	12	23	2.414	0.520	2	0.448	0.672	0.538
smell	3	7	3	73	230	5.450	0.371	4	0.519	0.546	0.532
sort	3	4	5	90	202	12.435	0.559	2	0.605	0.953	0.740
source	3	10	3	16	30	2.680	0.492	4	0.672	0.547	0.603
suspend	3	9	4	78	211	7.069	0.448	4	0.391	0.883	0.542
talk	8	12	4	35	89	4.138	0.450	3	0.808	0.733	0.769
treat	3	9	3	49	149	4.347	0.301	6	0.438	0.455	0.446
use	3	6	2	6	10	1.183	0.620	1	0.731	0.692	0.711
wash	12	7	3	26	47	4.346	0.534	3	0.576	0.652	0.611
watch	3	8	3	41	122	3.297	0.350	3	0.750	0.870	0.806
win	16	9	3	31	72	4.025	0.380	5	0.603	0.436	0.506
write	3	14	3	7	12	1.270	0.604	1	0.455	0.886	0.601

Table 5.11: Results of *HyperLex_OP+KPP* for All Target Words (2)

It is found that the number of the hubs in *HyperLex_OP+SYN* tends to be smaller than *HyperLex_OP*.

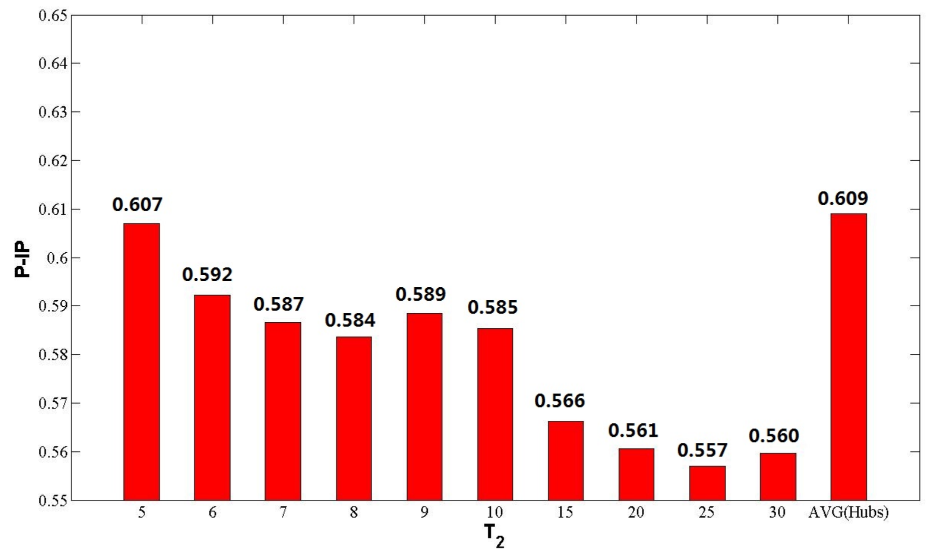


Figure 5.3: Results of all *HyperLex_OP+KPP*

Figure 5.3 compares the performance of *HyperLex_OP+KPP* for different threshold T_2. As in Figure 5.2, the figure shows the average of $P\text{-}IP$ for all words. $T_2 = AVG(Hubs)$ achieved the best among various T_2. Therefore, the heuristics to determine T_2 as the average number of the hubs for various parameter combination worked well.

WSI Method	**Hub**	P	IP	$P\text{-}IP$
Baseline	21.8	0.698	0.410	0.482
HyperLex_OP	14.6	0.639	0.469	0.513
HyperLex_OP+KPP	4.21	0.601	0.680	0.609

Table 5.12: Comparison among Baseline, *HyperLex_OP* and *HyperLex_OP+KPP*

Table 5.12 compares the baseline, *HyperLex_OP* and *HyperLex_OP+KPP*. T_2 is set to $AVG(Hubs)$ in both *HyperLex_OP* and *HyperLex_OP+KPP*. $P\text{-}IP$ of *HyperLex_OP+KPP* is much better than the baseline and *HyperLex_OP*. One of the important characteristics of *HyperLex_OP+KPP* is that the number of hubs or induced senses are much smaller. It causes great improvement of Inverse Purity at the cost of Purity. On average, however, $P\text{-}IP$ is improved. Therefore, the method choosing the hubs with graph connectivity measure is effective.

5.5.3 Results of Weighting Scheme Considering Syntactic Relation

Finally, we evaluate our weighting scheme where both co-occurrence of words and syntactic relations between words are under consideration. *HyperLex_OP+KPP+SYN* stands for an extension of *HyperLex_OP+KPP*: wegiths of the edges are determined by not only co-occurrence probability but also the number of words under the syntactic relations as Equation (4.11). Table 5.13 and 5.14 stand for the results of*HyperLex_OP+KPP+SYN* ($T_2 = AVG(Hubs)$) for all target words.

Target Word	P_1	P_2	P_3	N	E	$\frac{C}{C_{rand}}$	$\frac{L}{L_{rand}} - 1$	Hub	P	IP	$P\text{-}IP$
activate	3	18	6	46	93	8.543	0.535	5	0.811	0.662	0.729
add	3	14	6	42	72	9.024	0.667	2	0.548	0.939	0.692
appear	3	9	6	92	215	11.669	0.506	5	0.592	0.585	0.589
argument	15	12	5	121	309	13.996	0.461	10	0.507	0.774	0.612
arm	3	12	6	107	365	6.700	0.309	8	0.771	0.489	0.598
ask	16	11	5	58	116	8.368	0.580	5	0.525	0.418	0.465
atmosphere	16	10	4	79	247	6.459	0.356	5	0.497	0.708	0.584
audience	3	4	7	67	139	8.239	0.522	5	0.740	0.700	0.719
bank	3	14	8	81	174	10.774	0.547	5	0.653	0.821	0.727
begin	3	14	7	29	60	4.409	0.534	2	0.519	0.950	0.672
climb	12	12	4	31	58	5.151	0.520	6	0.556	0.383	0.454
decide	15	9	4	38	73	6.094	0.527	4	0.779	0.762	0.770
degree	3	11	6	114	274	13.637	0.490	7	0.641	0.633	0.637
difference	3	7	5	134	336	13.554	0.501	7	0.473	0.845	0.607
different	3	7	4	43	75	6.801	0.616	2	0.520	0.939	0.670
difficulty	3	7	3	27	79	1.638	0.313	3	0.500	0.826	0.623
disc	3	4	7	77	149	9.923	0.651	2	0.410	0.965	0.575
eat	3	9	5	54	113	7.055	0.577	3	0.845	0.923	0.882
encounter	3	11	5	34	60	5.632	0.616	2	0.592	0.592	0.592
expect	11	11	4	45	111	5.921	0.457	5	0.744	0.840	0.789
express	3	11	4	49	107	6.982	0.465	9	0.655	0.264	0.376
hear	3	9	2	14	35	1.161	0.461	2	0.508	0.905	0.651
hot	3	10	3	14	23	2.624	0.477	3	0.826	0.523	0.641
image	3	7	5	105	304	7.300	0.279	5	0.596	0.623	0.609
important	3	6	3	28	66	2.701	0.501	1	0.417	0.972	0.583
interest	3	15	6	36	64	6.756	0.598	4	0.384	0.600	0.468
judgment	3	7	4	25	42	3.945	0.662	1	0.419	0.903	0.573
lose	3	8	3	34	90	2.727	0.334	4	0.507	0.577	0.540

Table 5.13: Results of *HyperLex_OP+KPP+SYN* for All Target Words (1)

Target Word	P_1	P_2	P_3	N	E	$\frac{C}{C_{rand}}$	$\frac{L}{L_{rand}} - 1$	Hub	P	IP	$P\text{-}IP$
mean	3	7	4	35	63	5.033	0.524	3	0.600	0.663	0.630
miss	3	8	4	14	23	2.504	0.549	2	0.466	0.655	0.544
note	3	9	4	75	170	10.147	0.538	4	0.530	0.955	0.682
operate	3	5	2	29	81	1.835	0.195	3	0.629	0.629	0.629
organization	3	8	4	116	379	7.415	0.311	9	0.786	0.321	0.456
paper	3	6	5	136	323	14.799	0.515	5	0.306	0.828	0.447
party	16	10	7	102	379	5.141	0.340	3	0.704	0.865	0.777
performance	3	9	5	81	199	8.469	0.497	6	0.366	0.570	0.446
plan	3	11	5	80	209	8.060	0.445	6	0.735	0.422	0.536
play	14	7	4	53	113	5.020	0.394	6	0.490	0.433	0.460
produce	3	15	5	44	94	7.093	0.519	6	0.618	0.441	0.515
provide	3	11	6	35	84	3.406	0.426	2	0.824	0.868	0.845
receive	3	10	4	19	34	3.388	0.527	2	0.846	0.654	0.738
remain	3	12	5	41	73	7.307	0.486	5	0.856	0.266	0.406
rule	3	12	4	16	30	2.620	0.556	2	0.797	0.797	0.797
shelter	14	9	5	87	168	12.566	0.547	11	0.500	0.362	0.420
simple	3	7	2	10	18	1.955	0.541	2	0.722	0.583	0.645
solid	3	9	2	12	23	2.346	0.520	2	0.448	0.672	0.538
smell	3	7	3	73	230	5.450	0.371	4	0.519	0.546	0.532
sort	3	4	5	90	202	12.123	0.559	2	0.605	0.953	0.740
source	3	10	3	16	30	2.455	0.490	4	0.688	0.563	0.619
suspend	3	9	4	78	211	6.985	0.448	4	0.391	0.914	0.547
talk	8	12	4	35	89	4.089	0.450	3	0.808	0.705	0.753
treat	3	9	3	49	149	4.323	0.301	6	0.464	0.509	0.486
use	3	6	2	6	10	1.183	0.620	1	0.731	0.692	0.711
wash	12	7	3	26	47	4.137	0.544	3	0.576	0.621	0.598
watch	3	8	3	41	122	3.236	0.350	3	0.750	0.870	0.806
win	16	9	3	31	72	3.865	0.380	5	0.628	0.449	0.524
write	3	14	3	7	12	1.175	0.604	1	0.455	0.886	0.601

Table 5.14: Results of *HyperLex_OP+KPP+SYN* for All Target Words (2)

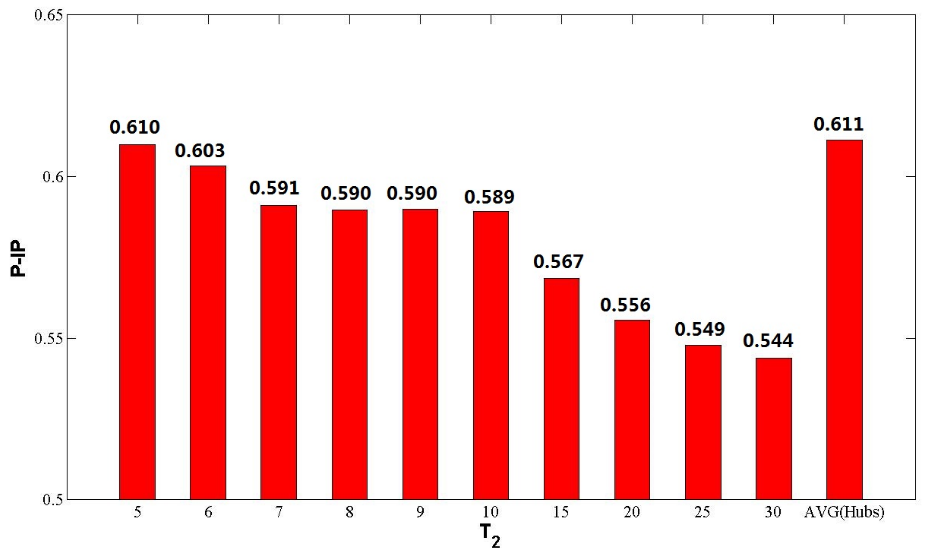


Figure 5.4: Results of all *HyperLex_OP+KPP+SYN*

Figure 5.4 shows the performance of *HyperLex_OP+KPP+SYN* for different threshold T_2. Similar to Figure 5.2 and 5.3. $P\text{-}IP$ tends to be small when T_2 is set high. Although $T_2 = 5$ and $AVG(Hubs)$ are comparable, $T_2 = AVG(Hubs)$ is the best threshold.

WSI Method	**Hub**	P	IP	$P\text{-}IP$
Baseline	21.8	0.698	0.410	0.482
HyperLex_OP	14.6	0.639	0.469	0.513
HyperLex_OP+KPP	4.21	0.601	0.680	0.609
HyerLex_OP+KPP+SYN	4.06	0.603	0.681	0.611

Table 5.15: Comparison of *HyperLex_OP+KPP+SYN* and Other Systems

Table 5.15 compares the baseline, *HyperLex_OP*, *HyperLex_OP+KPP* and *HyperLex_OP+KPP+SYN*. Note that T_2 is set to $AVG(Hubs)$ in all systems except for the baseline. We focus on the comparison between the system with and without considering syntactic relations for weighting. *HyperLex_OP+KPP+SYN* outperforms *HyperLex_OP+KPP* for Purity, Inverse Purity and $P\text{-}IP$. However, the improvement is not so great than we have expected. The reason may be that the weights of the edges are used for delineating components and disambiguation of the sense of a new sentence, but not used for construction of the graph. In other words, even when the weights are changed by considering syntactic relations, the structure of the graph is not changed. In addition, the average number of the hubs or senses of *HyperLex_OP+KPP+SYN* is slightly less than *HyperLex_OP+KPP*, although the structures of the co-occurrence graphs in these systems are same.

Chapter 6

Conclusion

6.1 Summary

This thesis presented a graph based word sense induction method that is an extenstion of HyperLex. The co-occurrence graph where vertices correspond to words appearing in the context of the target word is constructed. Edges between vertices represent relations between words. Weights of the edges indicate how two words correlated. In this method, the weights of the edges are determined based on both co-occurrence and syntactic relations of words. Furthermore, the parameters that strongly influence the co-occurrence graph are optimized so that the graph meets the small world properties. Then the hubs in the graph are detected by Key Player Problem algorithm. Finally, the graph is subdivided into the small pieces of trees whose root node is a hub. Each sub-tree represents the induced sense of the target word. The performance of the proposed WSI method was evaluated on a sense tagged corpus for 57 target words. The results of the experiments showed that the proposed method outperformed the original HyperLex. Furthermore, the number of induced sense was more similar to the genuine number of senses.

We summarize our contributions as follows:

1. In HyperLex, there are three parameters, and these parameters are determined in ad-hoc manner. In this research, the optimization of the parameters are examined. To find the best parameters, various combinations of parameters will be tested until the graph meets the condition of small world properties: $L \sim L_{rand}$ and $C \gg C_{rand}$. Our method can determine the parameters appropriately for different sizes of corpora.

2. In order to analyze co-occurrence graph connectivity more precisely and divide the graph into better components, we use a graph connectivity measure - *Key Player Problem (KPP)*. The use of *KPP* enhanced the performance of WSI in our experiment.

3. A weighting scheme considering syntactic relations between words was introduced. We found that syntactic relations contributed the improvement of WSI.

6.2 Future Work

Results of the experiments have shown that *HyperLex_OP+KPP+SYN* achieved higher performance than the original HyerLex (for all values of the threshold T_2). However, comparing with *HyperLex_OP+KPP*, the Purity and Inverse Purity of *HyperLex_OP+KPP+SYN* were not so improved. In our method, syntactic relations are considered to determine the weight of edges, but not used for construction of the graph. In future, we will examine a method to the build co-occurrence graph considering the syntactic relations between two words to improve the performance of WSI.

Acknowledge

First of all, I would like to to express my heartfelt thanks to my advisor **Associate Professor Kiyoaki Shirai** of Japan Advanced Institute of Science and Technology for his kindly guidance and supports. This thesis would not have been possible unless I did consult by him. As my supervisor, he taught me a lot of things not only the knowledge about natural language processing but also the research methodology, developing the new idea, and solving problems.

Besides, I would like to thank **Professor Tojo** and **Associate Professor Nguyen** for their support. It is their advice and guidance in the mid-term defense so that I can complete my research.I also would like to thank the member in Shimazu & Shirai Laboratory of Japan Advanced Institute of Science and Technology. I received a lot of help from them.

I would like to thank my lovely family and friends who always support and cheer me up when I feel down.

Finally, I sincerely would like to thank **TJU-JAIST Program**. It gave me plenty of chances to improve my knowledge and research skill in Japan.

Bibliography

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

[2] David Pinto, Paolo Rosso, and Hector Jimenez-Salazar. UPV-SI: Word sense induction using self term expansion. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, pages 430–433. Association for Computational Linguistics, 2007.

[3] Amruta Purandare and Ted Pedersen. Word sense discrimination by clustering contexts in vector and similarity spaces. In *Proceedings of the Conference on Computational Natural Language Learning*, volume 72. Boston, 2004.

[4] Wesam Elshamy, Doina Caragea, and William H. Hsu. KSU KDD: Word sense induction by clustering in topic space. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 367–370, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[5] Tim Van de Cruys. Using three way data for word sense discrimination. In *COLING*, pages 929–936, 2008.

[6] Eneko Agirre, David Martínez, Oier López De Lacalle, and Aitor Soroa. Evaluating and optimizing the parameters of an unsupervised graph-based wsd algorithm. In *Proceedings of the first workshop on graph based methods for natural language processing*, pages 89–96. Association for Computational Linguistics, 2006.

[7] Eneko Agirre and Aitor Soroa. UBC-AS: A graph based unsupervised system for induction and classification. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, pages 346–349. Association for Computational Linguistics, 2007.

[8] Beate Dorow and Dominic Widdows. Discovering corpus-specific word senses. In *Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics-Volume 2*, pages 79–82. Association for Computational Linguistics, 2003.

[9] Ioannis P. Klapaftis and Suresh Manandhar. Word sense induction & disambiguation using hierarchical random graphs. In *Proceedings of the 2010 conference on*

empirical methods in natural language processing, pages 745–755. Association for Computational Linguistics, 2010.

[10] Ioannis P. Klapaftis and Suresh Manandhar. Word sense induction using graphs of collocations. In *ECAI*, pages 298–302, 2008.

[11] Chris Biemann. Chinese whispers: an efficient graph clustering algorithm and its application to natural language processing problems. In *Proceedings of the first workshop on graph based methods for natural language processing*, pages 73–80. Association for Computational Linguistics, 2006.

[12] Ioannis P. Klapaftis and Suresh Manandhar. UOY: a hypergraph model for word sense induction & disambiguation. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, pages 414–417. Association for Computational Linguistics, 2007.

[13] David Jurgens. Word sense induction by community detection. In *Proceedings of TextGraphs-6: Graph-based Methods for Natural Language Processing*, pages 24–28. Association for Computational Linguistics, 2011.

[14] Jey Han Lau, Paul Cook, Diana McCarthy, David Newman, and Timothy Baldwin. Word sense induction for novel sense detection. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 591–601. Association for Computational Linguistics, 2012.

[15] Ioannis Korkontzelos, Ioannis Klapaftis, and Suresh Manandhar. Graph connectivity measures for unsupervised parameter tuning of graph-based sense induction systems. In *Proceedings of the Workshop on Unsupervised and Minimally Supervised Learning of Lexical Semantics*, pages 36–44. Association for Computational Linguistics, 2009.

[16] Duncan J Watts and Steven H Strogatz. Collective dynamics of "small-world" networks. *nature*, 393(6684):440–442, 1998.

[17] R. Navigli and M. Lapata. An experimental study of graph connectivity for unsupervised word sense disambiguation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(4):678–692, April 2010.

[18] Stephen P. Borgatti. The key player problem1. In *Dynamic social network modeling and analysis: Workshop summary and papers*, page 241. National Academies Press, 2003.

[19] Roberto Navigli and Mirella Lapata. Graph connectivity measures for unsupervised word sense disambiguation. In *IJCAI*, pages 1683–1688, 2007.

[20] Enrique Amigó, Julio Gonzalo, Javier Artiles, and Felisa Verdejo. A comparison of extrinsic clustering evaluation metrics based on formal constraints. *Information retrieval*, 12(4):461–486, 2009.

[21] Javier Artiles, Julio Gonzalo, and Satoshi Sekine. The SemEval-2007 WePS evaluation: Establishing a benchmark for the web people search task. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, pages 64–69. Proceedings of the 4th International Workshop on Semantic Evaluations, 2007.