Word Sense Disambiguation: An Investigation into Mechanisms for Sense Discrimination

Project Report

Submitted in partial fulfillment of the requirements for the degree of

Bachelor of Technology

By

DIPTESH KANOJIA

Roll No: 0944910004

Under the guidance of

Prof. (Dr.) Pushpak Bhattacharyya

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

March 2013

Acknowledgement

I would like to express my profound gratitude to **Prof. (Dr.) Pushpak Bhattacharyya** for his *invaluable guidance* and support for rigorous study of literature. His essential and unparalleled inputs and suggestions during difficult phases were much required. I am highly indebted to him for his kind help. I thank my senior research fellows, **Salil Joshi** and **Arindam Chatterjee** for the minutely detailed discussions we had on Word Sense Disambiguation, and Human Annotation Techniques. I thank **Raj Dabre** for helping me out with PHP coding, whenever needed, and his immensely supportive nature during the course of this project. I would also like to thank our team of lexicographers **Jaya Ma'am**, **Rajita Ma'am** and **Nootan Ma'am** for manual clue acquisition during the course of this project. This project could not have been complete without their help.

I would also like to thank **Shri S.K. Jain (Chairman)**, K.S. Jain Institute of Engineering and Technology, for his constant moral support, and encouraging nature towards my research projects.

Finally, I would like to thank my dear friends **Nitin Kumar Singh, Ashish Sharma, Anu Gupta, Russell Quadros** and **Deck-16 people** for supporting me throughout all my tough times, and encouraging me to work harder every time I transgressed.

Diptesh Kanojia
(0944910004)

Abstract

Word Sense Disambiguation (WSD) is defined as the task of computationally finding the senses of words in a sentence. Our experiments in eye tracking have revealed that the words in the context of the word to be disambiguated are vital. As such, it becomes important to enumerate all possible clues beforehand so that they can be conveniently used at runtime. A tool which can allow for such annotation, manually, as well as utilize heuristics to generate clues automatically, is crucial.

We describe the sense discrimination tool constructed for the purpose of developing a "Discrimination Net" which is a resource that can be utilized for high accuracy word sense disambiguation. We go over the building blocks of the tool, describing its workflow. Thereafter, we delineate the process of automatic clue acquisition by means of extracting words from sentences returned by a concordancer, which is a tool that can allow one to search sentences containing the synset words from a corpus.

This is followed by a study of methods such as PMI (Pointwise Mutual Information) which will be used for the re-ranking of clues and filtering out irrelevant ones. We also give a case analysis of the performance of PMI to demonstrate its efficacy. Finally, a documentation of our tool is also incorporated. This tool is highly language independent and can be ported to other languages quite easily.

We imagine that constructing a "Discrimination Net" in the form of a weighted graph will assist in calculating a score which will help alleviate the feeling of uncertainty in determining word senses. The underlying idea is that there are words with multiple senses as well as ones with unique senses present and by traversing this graph, we will eventually reach these unique senses and then determine the score.

Contents

Chapter 1

Introduction

Major Natural Language Processing applications rely on Word Sense Disambiguation, which depends on sense tagged corpus. Sense tagging involves assignment of Part of Speech tag and WordNet Sense ID to the words in the document provided. Sense marker tool helps lexicographers in sense tagging or better known as human annotation. It assumes one sense per discourse for faster tagging, and also assumes stems of unique words in corpora are available in absence of stemmer.

1.1 Word Sense Disambiguation (WSD)

Word Sense Disambiguation (WSD) is one of the toughest problems today, not only in Natural Language Processing (NLP) but also in Artificial Intelligence (AI). The problem of WSD dates back to 1950s. It basically refers to the automatic disambiguation of word senses and it has been an interest and concern since the earliest days of treating languages computationally.

According to experts, sense disambiguation is an "intermediate task", which is just a stepping stone to most Natural Language Processing tasks. Before moving into further detail, let us find out what exactly WSD is, and the various ways of looking at the problem, along with the various forms of the problem and *why the problem is favorite among NLP researchers*.

1.1.1 WSD Definitions

Initially it is important to know the basic definitions of the WSD problem and its different types. The following section deals with the evolution of WSD as a problem. In light of the motivation behind WSD, let us have a look at the colloquial and formal definitions of WSD.

1.1.1.1 General Definition

WSD is the ability to identify the meaning of words in context in a computational manner. Given a set of words (*e.g.*, a sentence or a bag of words), a technique is applied which makes use of one or more sources of knowledge to associate the most appropriate senses with words in context. In light of the motivation behind WSD, let us have a look at the colloquial and formal definitions of WSD.

1.1.1.2 Formal definition

WSD is the task of assigning the appropriate sense(s) to all or some of the words in T, that is, to identify a mapping A from words to senses, such that $A(i) \in Senses_D(w_i)$, where $Senses_D(w_i)$ is the set of senses encoded in a dictionary D for word w_i, and A(i) is that subset of the senses of w_i which are appropriate in the context T. The mapping A can assign more than one sense to each word $w_i \in T$, although typically only the most appropriate sense is selected, that is, $|A(i)| = 1$. Where, T is a sequence of words $(w_1, w2, \ldots, w_n)$.

1.1.1.3 As a Classification Problem

Word senses are the classes, and an automatic classification method is used to assign each occurrence of a word to one or more classes based on the evidence from the context and from external knowledge sources. Other classification tasks are part-of-speech tagging, named entity resolution, text categorization, etc. But important difference between these tasks and WSD is that the former use a single predefined set of classes (parts of speech, categories, etc.), whereas in the latter the set of classes typically changes depending on the word to be classified.

1.1.2 Variants of WSD

The problem of Word Sense Disambiguation can be broadly classified into two categories.

1.1.2.1 Lexical Sample (Targeted WSD)

Here, the system is required to disambiguate a restricted set of target words usually occurring one per sentence. It employs supervised techniques using hand-labeled instances as training set and then an unlabeled test set.

1.1.2.2 All-words WSD

Here, the system is expected to disambiguate all open-class words in a text (i.e., nouns, verbs, adjectives, and adverbs) therefore, they are termed as wide coverage systems to disambiguate all open-class words. Typically, such a system suffers from Data sparseness problem, as large knowledge sources are not available.

1.1.3 Current Approaches

1.1.3.1 Dictionary and knowledge-based methods

These rely primarily on dictionaries, thesauri, and lexical knowledge bases, without using any corpus The Lesk algorithm is the seminal dictionary-based method. It is based on the hypothesis that words used together in text are related to each other and that the relation can be observed in the definitions of the words and their senses.

1.1.3.2 Supervised methods

These make use of sense-annotated corpora to train and are based on the assumption that the context can provide enough evidence on its own to disambiguate words (hence, world knowledge and reasoning are deemed unnecessary).

1.1.3.3 Semi-supervised or minimally supervised methods

These make use of a secondary source of knowledge such as a small annotated corpus as seed data in a bootstrapping process, or a word-It uses the 'One sense per collocation' and the 'One sense per discourse' properties of human languages for word sense disambiguation. From observation, words tend to exhibit only one sense in most given discourse and in a given collocation.

1.1.3.4 Unsupervised methods

These eschew (almost) completely external information and work directly from raw unannotated corpora. Here, the underlying assumption is that similar senses occur in similar contexts, and thus senses can be induced from text by clustering word occurrences using some measure of similarity of context, a task referred to as word sense induction or discrimination.

Almost all these approaches normally work by defining a window of n content words around each word to be disambiguated in the corpus, and statistically analyzing those n surrounding words. Two shallow approaches used to train and then disambiguate are **Naïve Bayes classifiers** and **decision trees**.

In recent research, kernel-based methods such as **support vector machines** have shown superior performance in supervised learning. Graph-based approaches have also gained much attention from the research community, and currently achieve performance close to the state of the art.

1.1.3.5 Other approaches

Other approaches may vary differently in their methods:

- Identification of dominant word senses.
- Domain-driven disambiguation.
- WSD using Cross-Lingual Evidence.
- Disambiguation based on operational semantics of default logic.

1.1.4 Current Limitations

Even Supervised Word Sense Disambiguation faces many challenges and there are many factors which limit the performance of this system to around 70% accuracy in the literature. Some of these factors are described below.

1.1.4.1 Definition of the problem

Some authors claim that defining the meaning of a word as a discrete list of senses is hopeless, as it does not model correctly its behavior.

1.1.4.2 Sense Inventory and Granularity

The task depends on the applied sense inventory, which has to be chosen adequately in order to build a flexible and comparable system.

1.1.4.3 Inadequate application of ML algorithms

Methods coming from the Machine Learning community have been widely applied to the WSD problem. However, the comparative results show that even the most sophisticated methods have not been able to make a qualitative jump in performance.

1.1.4.4 Limited Feature sets

Traditionally simple feature sets consisting in bigrams, trigrams and “bags of words” have been used to model the contexts of target words. But in order to be robust, ML methods should rely in as much information from the texts as possible.

1.1.4.5 Sparse Data

In NLP most of the events occur rarely, even when large quantities of data are available. This problem is specifically noticeable in WSD, where hand-tagged data is difficult to obtain.

1.1.4.6 Need of extra training data

Existing manually tagged corpora is not enough for current state-of-the-art systems. Hand-tagged data is difficult and costly to obtain, and methods to obtain data automatically have not reached the same quality of hand-tagged data so far.

1.1.4.7 Portability

The porting of the WSD systems to be tested on a different corpora than one used for training also presents difficulties. Previous works (Ecudero, Marquez, Y Rigau, 2000) show that there is a loss of performance when training on one corpora and testing on another.

1.1.5 WSD: Heart of the NLP Problems

As mentioned earlier, due its applicability and hardness, WSD has become an area of keen interest and challenge for NLP researchers. Word Sense Disambiguation is truly the heart of all NLP problems. This very fact is illustrated in the figure below:

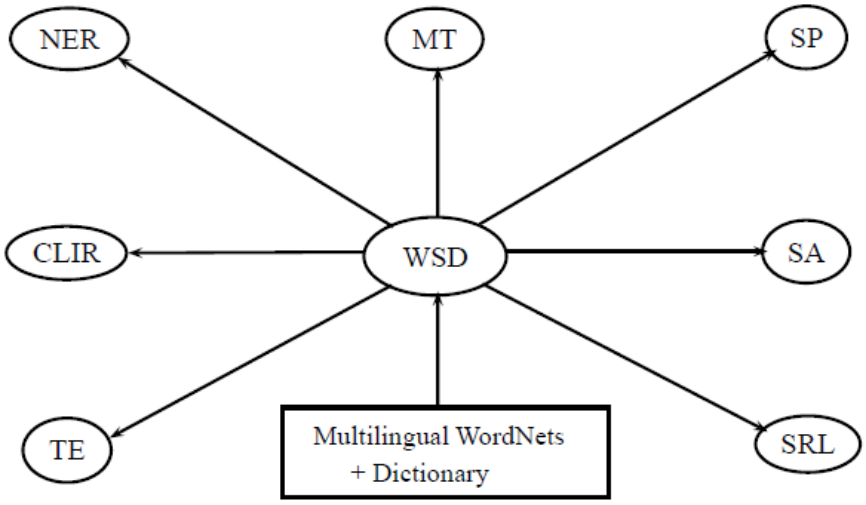


Figure 1.1: WSD - Heart of the NLP problems

- **SRL:** Semantic Role Labeling
- **TE:** Text Entailment
- **CLIR:** Cross Lingual Information Retrieval
- **NER:** Named Entity Recognition
- **MT:** Machine Translation
- **SP:** Shallow Parsing
- **SA:** Sentiment Analysis
- **WSD:** Word Sense Disambiguation

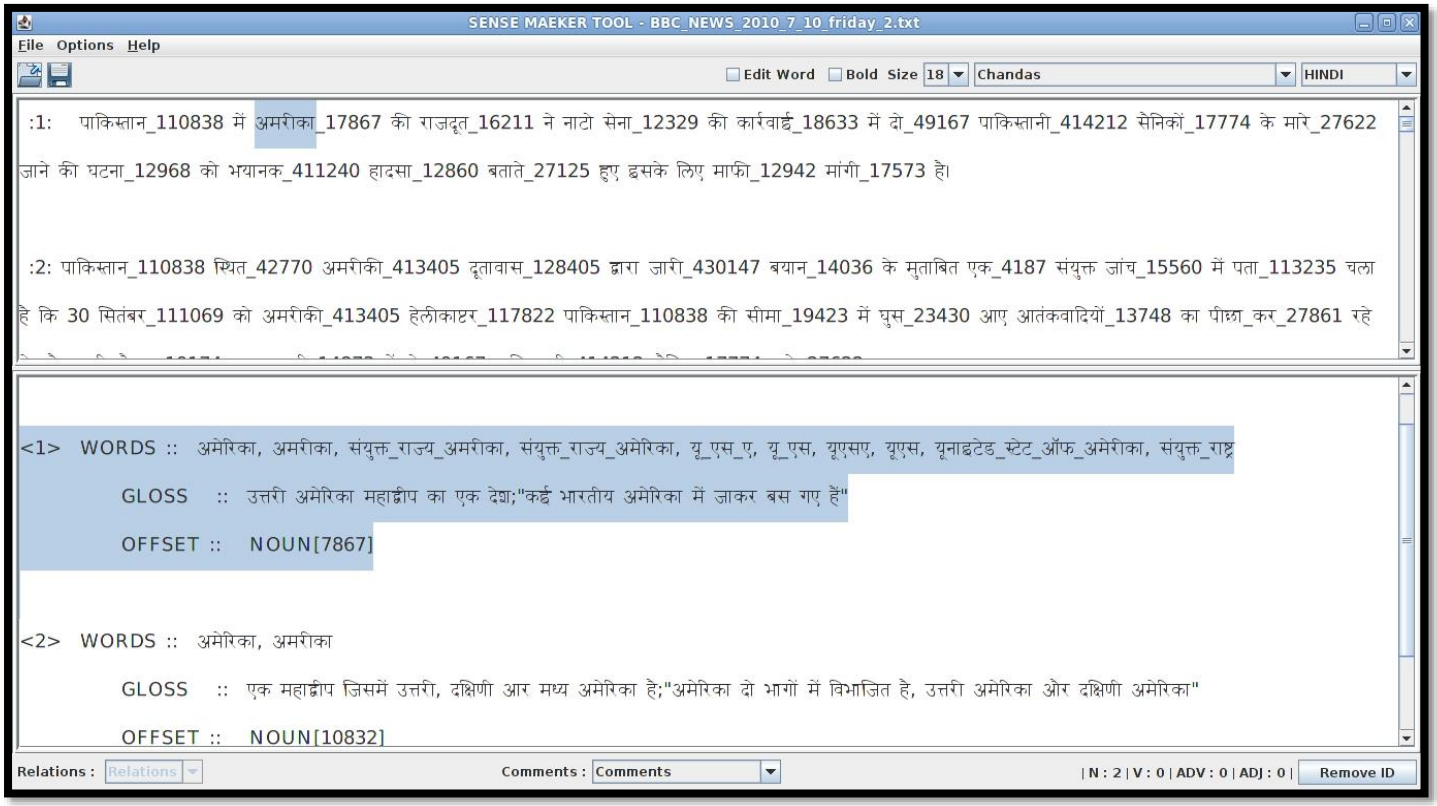
1.2 The Sense Annotation Process

In this section, we have demonstrated the backbone of our human annotation process, the sense marker tool. It serves as one single weapon to which helps our annotators mark the words with sense ids in 18 different languages. The machine needs to be trained by humans to understand the written language. Huge amount of accurately sense-marked data is supplied to the algorithm for its training.

1.2.1 The Sense Marker Tool

A word may have a number of senses and to identify and mark which particular sense has been used in the given context is known as sense marking.

At IIT Bombay, this work is being done in 3 languages – English, Hindi and Marathi. The corpus used so far have been taken from Tourism, Health, Environment and Travel review domains and the Princeton WordNet is used as the sense inventory for English text while the Hindi and Marathi WordNets have been used for Hindi and Marathi texts respectively.
The sense-marker tool developed by IITB supports 18 languages (English, Hindi, Marathi, Assamese, Bengali, Bodo, Gujarati, Kannada, Kashmiri, Konkani, Malayalam, Manipuri, Nepali, Oriya, Punjabi, Sanskrit, Telugu and Urdu).



Picture 1.2: The Sense Marker tool screenshot.

The Sense Marker tool is a Graphical User Interface based tool developed using Java which facilitates the task of manual sense marking. This tool displays the senses of the word as available in the Marathi, Hindi and Princeton (English) WordNets and allows the user to select the correct sense of the word from the candidate senses.

The table shown alongside is the statistics of sense-marking done for Tourism and Health files (for two languages: Marathi & Hindi):

Domain	Total Documents	Total Sentences	Tagged Words
Tourism Hindi	152	15,200	1,80,525
Tourism Marathi	152	15,200	1,25,387
Health Hindi	89	8,900	94,209
Health Marathi	72	7,200	51,415
Tourism English	152	15,200	1,81,964
Health English	140	14,000	1,49,259
Total	757	75,700	7,82,759

Table 1.1: *Statistics of sense marking*

1.3 Report Outline

This is an overview of the work done earlier, clearly there is a need for the exploration of new mechanisms for discrimination of senses. This report describes our "Sense Discrimination Tool", improvements and development of a clue suggestion mechanism based on raking of probable senses. Process of clue marking for sense discrimination is crucial for betterment of WSD systems, and in this work, we have made it much simpler. We have also laid groundwork for future automatic clue marking, and sense tagging. Chapter 2 outlines our vision for future and possible mechanisms and establishes solid theory on discrimination net. Chapter 3 gives detailed explanation of how we went on to develop our tool, and the hurdles which came across during it. Chapter 4 is built around the current work done which helped us reach the current phase of the tool. It encompasses the work done on possible clue suggestion and ranking them according to Point-wise mutual information (PMI) between the target word and the possible clue word. Chapter 5 contains the demonstration of our tools functionalities. Chapter 6 lists the results of study and conclusion drawn by it. Chapter 7 has our future plans regarding automation and development of discrimination net.

Chapter 2

Previous work & Overview

2.1 Human *vs.* Machine Annotation

The process of sense annotation of words with senses is more accurate for humans than machines. The deciding parameter in the human sense disambiguation process is contextual evidence. Considering the principle of *weak AI*, the annotation procedure employed by the machine should make use of contextual evidence for disambiguation purposes in some form, which also conforms to the classical definition of WSD.

We had established that **contextual evidence** is a *necessary attribute* for the human tagging process. Without contextual information, the human tagging process is crippled (Arindam Chatterjee, 2012). Machines, which use the *P(S|W)* statistic for WSD, take human context-sensitive information to learn the *P(S|W)* measure. This is an adaptation of the contextual evidence used by human beings. Hence the principle of *weak AI* holds for such WSD algorithms. Hence obtaining the *P(S|W)* values perfectly is of paramount concern for machines.

Supervised approaches to WSD deliver far better results, compared to knowledge-based or unsupervised methods (Navigli, Word Sense Disambiguation: A Survey, 2009). In a supervised framework, WSD is considered as a classification task, where senses of words are the classes. If we take a closer look at the state-of-the-art supervised algorithms for WSD, it will be evident that the parameters used by such algorithms are mostly statistical, *i.e.*, *corpus-based evidence*.

2.1.1 Recent Developments

Our earlier work had presented the variance in difficulty levels of annotation across various POS and ontological categories (Joshi et. Al, 2013), as recorded by the eye tracking device. The results in most cases were not stochastic, and conformed to the view of lexicographers. The second phase of the experiment, was difficult to analyze from the data recorded by the eye tracking device.

It formed a solid base for a future rule based framework, which would be self-sufficient or rather independent framework. It would be able to discriminate between polysemous word senses on its own, using context word set. We had discussed about developing a tool that would track contextual clues during annotation of a word with a particular sense would record the clue words aiding in the disambiguation from the lexicographers.

We successfully developed such a tool, and urged lexicographers to enter clues related to synset words. The tool works with a stored WordNet database and appends clues manually entered in it to the related synset. This completed the first stage of our tool development and we could now begin gathering clues for various synsets further helping us with disambiguation.

Using this we hope to develop a framework which uses the Discrimination Net constructed from the clue marked content, which would help us accurately disambiguate polysemous words, using the context word set.

2.2 Basics for manual annotation

The lexicographers began adding clues from the provided gloss and example fields containing the context words related to the synset id. We realized later that many other words which can be used as clues were not present in a single example sentence. We were motivated to providing our lexicographers with a rich corpus based result, which could help them in locating such word entries. We came up with the idea of adding concordancer search to our tool, incorporating huge results from a Hindi corpus, facilitating the addition of much more and better clues to aid disambiguation.

2.3 Providing Blind clues from the corpus

We wanted to automate the searching of clues from the concordancer as well and hence came up with a mechanism which would provide blind clues to the lexicographers based on the corpus search via POS tagging. We thought of using a hybrid CRF based POS tagger which does stemming and POS tags the words provided to it. Hence we used the following mechanism:

- For every sentence returned from the concordancer, we performed POS tagging using a CRF based hybrid POS tagger.
- Assuming from past experiences, we knew for a fact that neighboring nouns and verbs would act as good clues for the target word.
- We took all the Nouns and Verbs POS tagged in the sentences returned from the concordance search for the first ten sentences, and added them to be displayed on the page as possible candidates for clues.
- We then found a lot of Auxiliary verbs in the list, which could never be potential clues, hence we used a filtering list to filter all the Hindi stop words and function words from the list.
- We added auxiliary verbs and numerals to such a list and some punctuation marks as well, so that the possible clues box looks clean of all such characters and words, not needed by our lexicographers.

2.4 Devising heuristics to Re-Rank, Re-Order, and Revise

The clues being displayed in the possible clues text box on the page were helpful to the lexicographers but were not ordered nor sorted by their importance or relatedness. We thought of using measures to calculate the relatedness or similarity of the clues and then re-rank them and be displayed in such a fashion that the more helpful clues are on the top. On the suggestion of using point-wise mutual information (PMI) as a ranking index, we calculated the PMI between the target word and every possible clue candidate in the box, and obtained the results.

The results were good, since the relatedness of the two words was successfully being reflected in the output, hence we now had a successful mechanism for re-ranking of the possible clues. We also took to thought, the possibility of relevance of the clues with high PMI scores and ranked them based on their number of occurrences in the corpus as well, we used LOG10 and SQUARE ROOT functions to re order the clues and obtained interesting results for many such words.

We revised our list of possible clue words and now we had 3 separate ways of obtaining important clues from the possible candidates.

2.5 Automatically adding them to the clue set

We envision the automatic addition of the clue words to the database after such heuristics being applied to the clues in future. This will not only help the lexicographers but also start providing us an automatically clue generating tool for the whole database. We think such a mechanism will lead to complete automation of clue generation and will help the lexicographers save a lot of time. Only manual verification of such clues will then have to be done by them, and with such a final database, we would easily be able to generate a related word network leading the discrimination net.

2.6 Devise a mechanism to fully automate clue acquisition

We target in near future a mechanism which will help us acquire the clues automatically, and does not need any kind of verification. We need to develop a clue acquisition mechanism which acquires the clues from a large back end corpus and automatically ranks and adds them to the database, for future use.

2.7 Develop a discrimination net

Our current approach has lead us to automation of possible clue suggestion to the lexicographers and we tend to improve this process further and add the clues automatically to the database. After successfully completing the above mentioned task, we plan to use this clue set to relate the synset ids with them and develop a tool, which on input of any sentence would be able to disambiguate it with a very high accuracy. Possibly many other measures of relatedness will have to be taken into account before such a tool can be achieved, but we plan to achieve high accuracy through these context based and rule based mechanisms combined.

2.8 Projecting the Discrimination net onto other languages

We ultimately aim to project our concept of context based measures onto other languages as well. We aim to achieve high accuracies in disambiguating in other languages later by projecting this concept. We think there is high possibility of generalization of such a concept and this could very well be used to discrimination of sense ids and disambiguation of polysemous words in other languages as well.

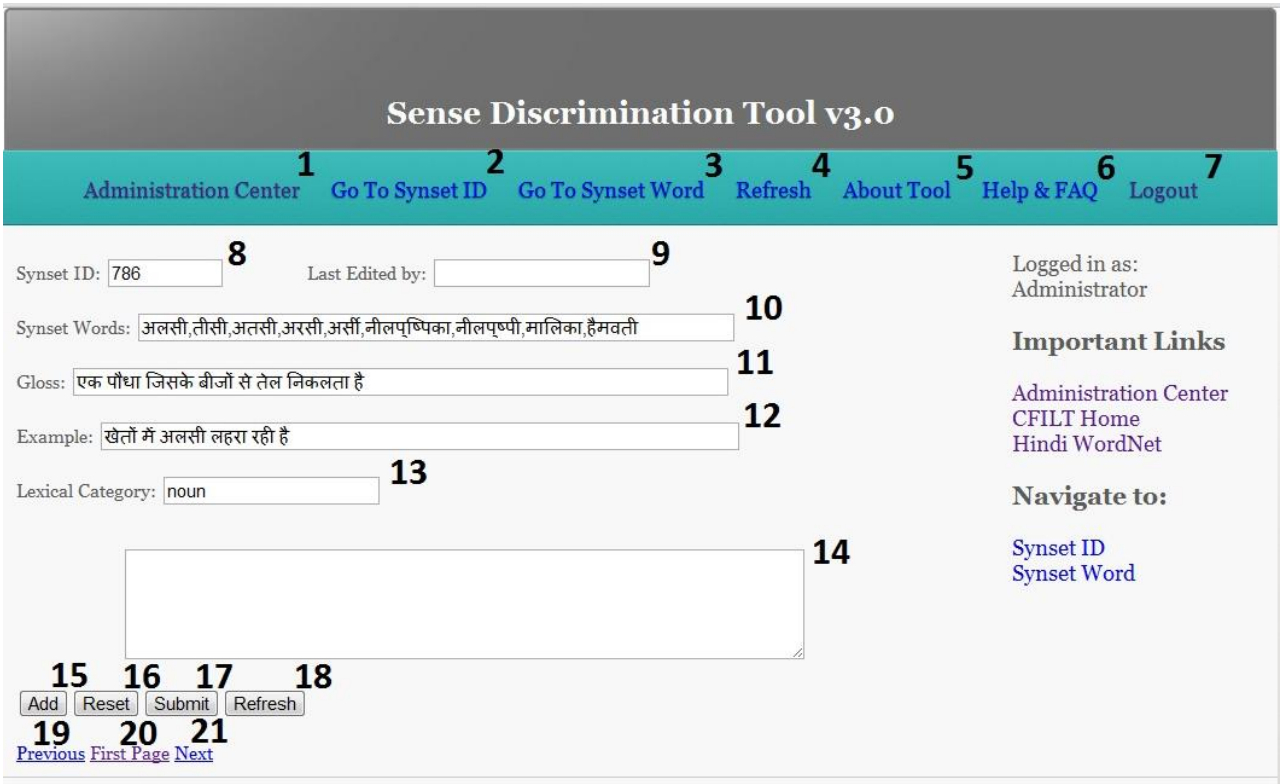
Chapter 3

Genesis

This chapter contains the path which led us to the current phase of our tool and lists the hurdles and failures which came across our way during the development of the tool. This contains a part by part analysis of how we made the tool, and our thought while coming out with this development.

Here we present an overview of the tool, and a detailed explanation of its previous interface. In the previous phase, we tended to collect clues only from the gloss and example fields already inculcated in the Hindi WordNet. In its previous phase "Sense Discrimination Tool" added manually entered sense clues to the database.

Given below is the detailed screenshot of the tool.



The Sense Discrimination Tool home page is shown above and it is described below:

1. **Administration Center**: For administrative users, Operations such as Approve, Reject, Ban and Super User and Delete user are present for an Administrator.
2. **Go To Synset ID**: Navigates to a particular Synset ID.
3. **Go To Synset Word**: Navigates to a particular Synset word, based on choice of user.

4. **Refresh**: Refreshes the page for showing updated clue words.
5. **About**: Opens a page explaining the tool
6. **Help**: Opens a page on how to use the tool, and who should use the tool.
7. **Logout**: Logs a user out.

Main Window:

8. Displays the **synset ID**.
9. Displays the **user id of the user who last edited the clue words** for this synset.
10. Displays the **Synset Words**.
11. Displays the **gloss** of the word in Hindi WN.
12. Displays the **example** of the word usage present in Hindi WN.
13. Displays the **Lexical Category** of the word.
14. **Clues text box** where user keeps adding the clues and can edit them before final submission.
15. **Adds the selected text** on the page to the Clues Text box[14].
16. **Resets the Clues Text Box[14]** for fresh clues addition.
17. **Submits the final entries in the Clues Text Box** in the database.
18. Refreshes the page.
19. Navigates to **Previous Synset** ID.
20. Navigates Back to **First Synset** ID.
21. Navigates to **Next Synset** ID.

3.1 How to use "Sense Discrimination Tool"?

Step One:

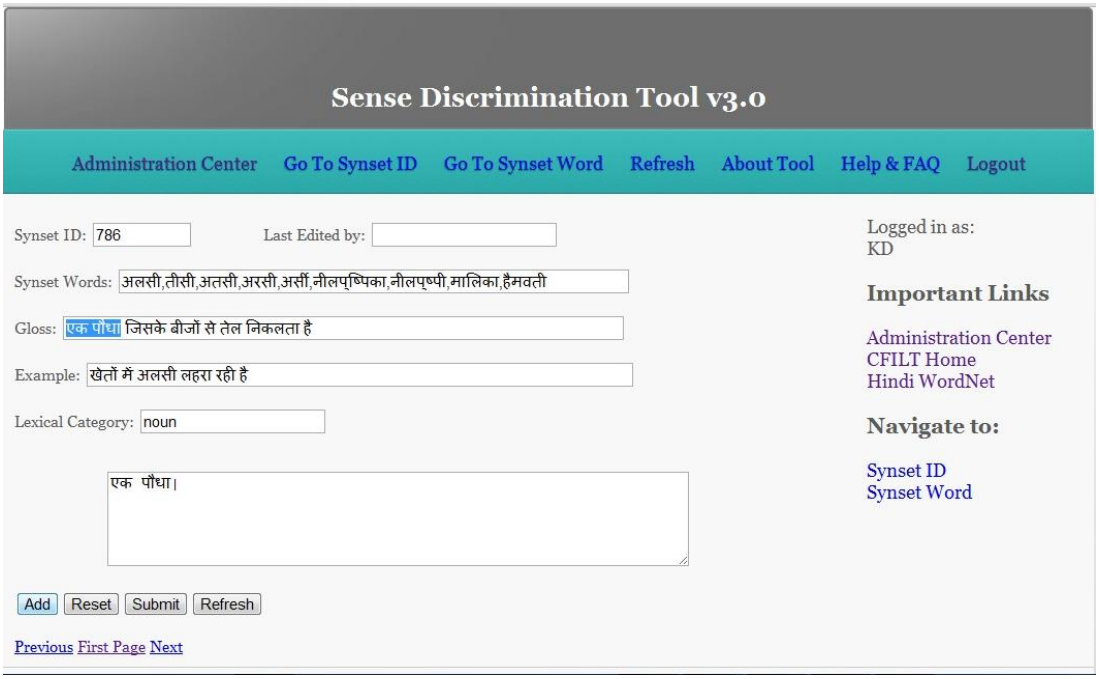
If you do not have a login ID and password, kindly create one by going to the Registration Page by clicking on the Create Login button on the Login Page.

Your ID has to be approved by the CFILT SysAd for a valid Email ID and only after approval you can login to work with the tool.

Step Two:

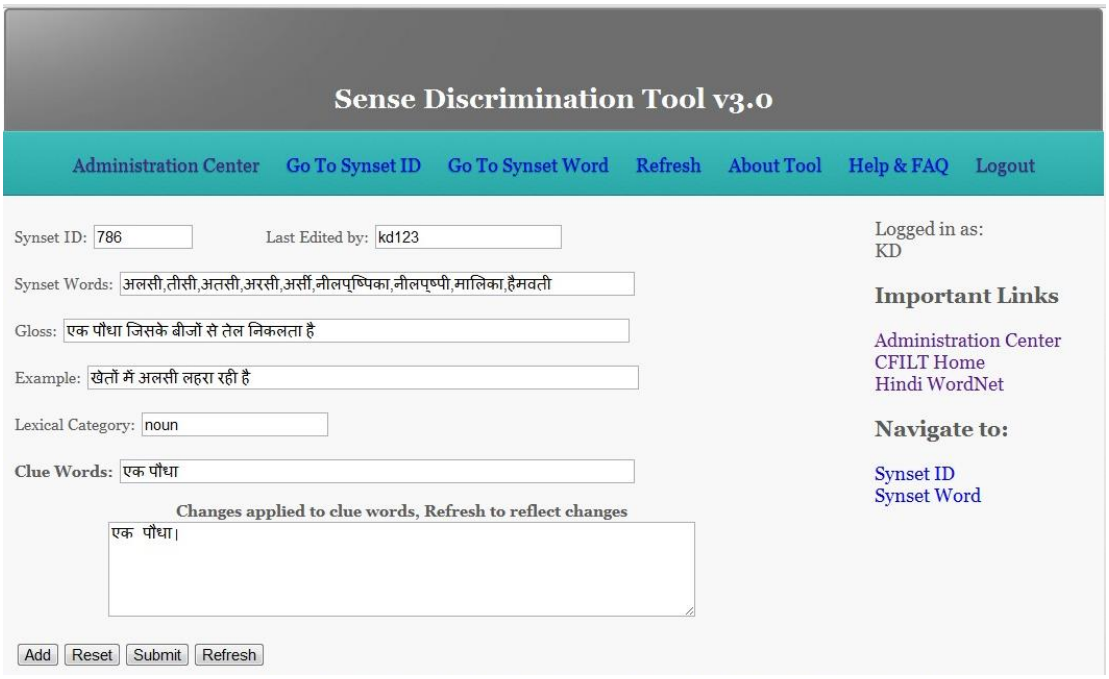
Once you have been approved, Login using your credentials and you will be taken to the tool Home Page show above.

Step 3: You have to identify the synset word in Synset Words[10] and select the word/phrase which you think helps disambiguate the word meaning and leads to the winner sense. Select that word/phrase using mouse or using SHIFT key on the keyboard. The clues will be available in gloss[12] and example[13].



Step 4: Now, Click on add to add them to the Clues Text Box[14] and edit them for any changes, if needed. Make the clues set final for addition to database.

Step 5: Click on Submit to add the phrases to the database.



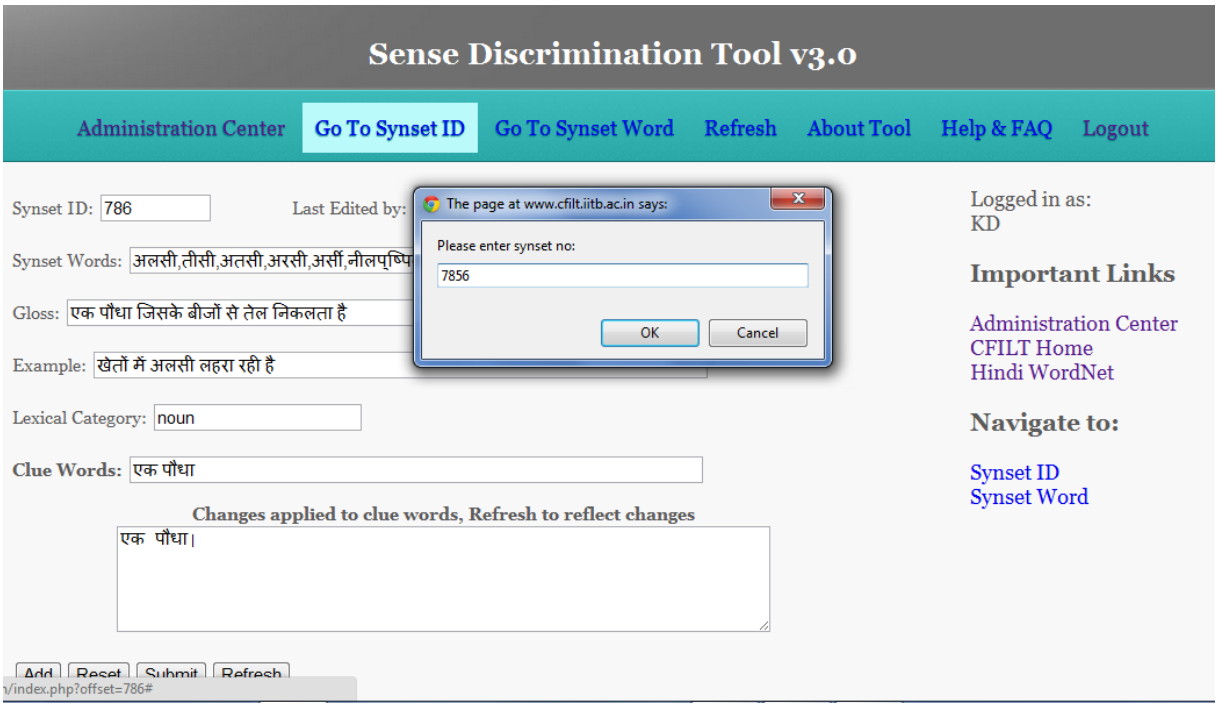
Step 6: After Submission, a new Text Field containing the added clue words will appear on the page. Navigate[2/3] to the next Synset ID, or to any Synset ID / Word as per your choice.

Step 7: After finishing, Click Logout[7] on the top right corner.

3.2 Navigation in Sense Discrimination Tool

3.2.1 Using "Go To Synset ID"

Clicking on Go To Synset ID in the navigation bar on top or in the sidebar, will pop up a small windows asking for target Synset ID.



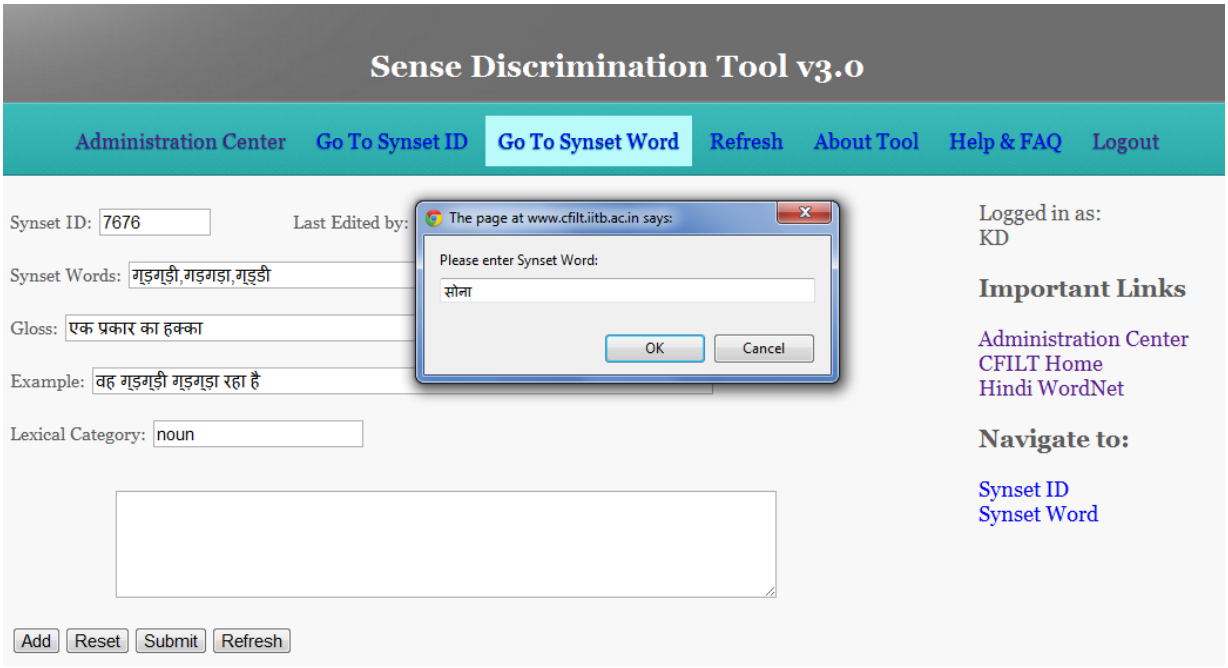
Enter the synset ID number in the text box pop up and click on OK to navigate to the particular Synset ID.

3.2.2 Using "Go To Synset Word"

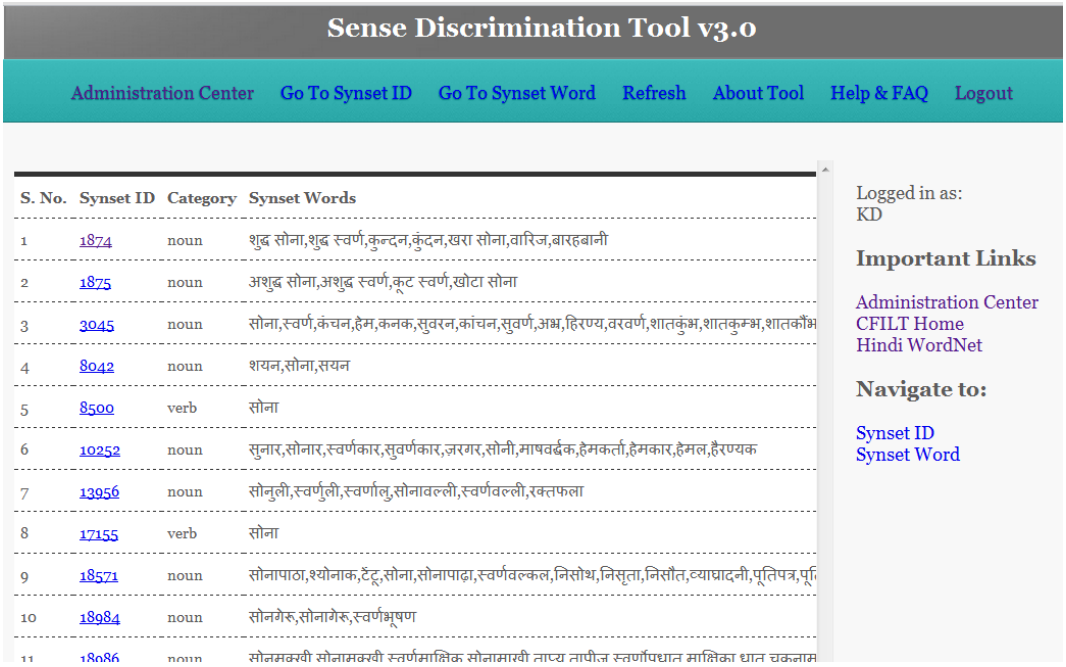
Clicking on Go To Synset Word in the navigation bar on top or in the sidebar, will pop up a small windows asking for target Synset Word. You can type in the synset word by changing your typing alternative to Hindi, or you can copy a Hindi word / its part from the Hindi WordNet.

For Convenience, Hindi WN link is provided in the sidebar.

The screenshot including the text box pop up is shown below:



After entering the word in the text box, Click OK or Press Enter. The navigation will take you to a page where all the resulting instances of the input word in the Hindi WN database are present.



From here, you can click on the target synset word you want to go to and the tool will open the page on that particular synset ID.

3.3 Initial Failures

3.3.1 File backed system

Initial development of the tool began with a miniature version which would pick up the synset words and related words from a binary file imported from the WordNet database. This version added clues to the file itself and was thought to be highly portable since the binary file could be copied anywhere. But this was scrapped as it presented with a lot of maintenance complexities and relation based complexities to the whole database.

The whole database was then shifted to MySQL and a lot of PHP coding issues were simplified then and there itself.

3.3.2 Manual Copying of clues

Initially the word selection which was made in the page had to be manually copy pasted in the clue box, which was wasting precious time of our lexicographers. Hence we facilitated the tool with an "Add" button which helped put any selected text in the page into the clue box immediately and added the pipe separator after it. This saves a lot of our time lost in copying, pasting and manually adding the separator between two clues.

This button uses a JavaScript function which was made completely browser independent and which helped get the selected text into the clue box with the separator at the end.

3.3.3 Simple login system Vs. A login system based on user precedence

Initially the login system being used in our tool was simple and only used to authenticate the people who could add clues to the database, but a record had to be kept as to who was the last user to modify the clues in the database, and precedence had to be given such that, clues being added by unskilled annotators will be given less importance.

Our skilled lexicographers were made super users, two admins of the system were created and a better login system based on user precedence was provided. This system also featured and administration center, visible only to the admins of the tool.

Chapter 4

Evolution

This chapter contains our current work and the most recent developments in the "Sense Discrimination Tool". As of now, the tool is up and running and allowing lexicographers to easily collect clues to be added to the database.

4.1 Sowing seeds (Initial Changes)

The previous phase of the tool allowed the lexicographers to collect the clues from the gloss and example fields and expected them to add other clues on their own regard. We found yet again that searching for clues on the web might waste a lot of precious time and hence some changes regarding this had to be made to the tool.

4.1.1 Concordancer Search

Hindi Concordancer was developed by Ashish Almeida under the guidance of Prof. Pushpak Bhattacharyya in 2004, and was available with the CFILT resources section. We took the database that it supported and facilitated a Concordancer search on our page. It allowed the lexicographers to look at the sentences containing the target word submitted to it. The concordancer consisted of 198000 (approx.) sentences initially which was already quite big a corpus.

We also added the NEWS domain corpus to it, further strengthening it and making it outputs more general in purpose. We cleaned the already sense tagged NEWS domain corpora and added it to the concordance database and now this database acts as a back end for the concordancer search available on our page. Its working will be shown in the coming chapters. Our concordancer search is Hindi transliteration enabled.

4.1.2 Google API for Hindi transliteration

Initially the tool clue box served the purpose of clue addition only through the external addition of clues either by copy pasting, or by selecting text on the page. Clues also needed to be manually edited sometimes so that their root from can be derived from the inflicted from in the clue box itself. We added the Google Transliteration API provided to us by Google on our tool page which enables the manual writing of Hindi words or sentence in the clue box itself. This enables us to Add/Edit clues in the box manually, thus saving time yet again.

The API also facilitated the typing of straight forward Hindi words in the concordancer box as well. The Concordancer search is Transliteration enabled and words can be typed into it directly.

4.1.3 Easy addition from Concordancer

The concordancer results which were being populated after the concordancer search were being shown in a separate new tab, from which the lexicographers had to manually copy paste clues to our clue box in the tool. This was rectified by another suggestion and now the results were being populated on the same page itself, facilitating the clue addition by a simple "Add to Clues" button below the results.

The results when shown on the same page make it easier and less time consuming for our lexicographers to extract clues from them and add them to the database. We tend to make clue acquisition automatic after all, but in current stage the least we can do is to make the task easier for our lexicographers.

4.2 Nurturing the saplings (Mining for possible clues)

4.2.1 CRF Based hybrid POS tagger

We knew for a fact that adjacent Nouns and Verbs usually for a good set of possible clues for almost all the target words. Hence, we took this thought and tried to suggest possible candidate clues to our lexicographers. We came up with a mechanism which would search the concordancer for the target word and out of the first ten sentences it will collect all the Nouns and Verbs.

For collecting all the Nouns and Verbs, POS tagging of the sentence was required, hence we used a CRF based hybrid POS tagger available with us, and used it to POS tag the sentences in the output from the concordancer search.

4.2.2 Filtration

Many of the auxiliary verbs which were being tagged as main verbs by the concordancer had to filter from the list, since they would not be useful clues in our lexicographers regard. We prepared a list of Hindi Stop words, Function words, Numerals and added such auxiliary verbs to it and filtered the possible set of clues before displaying them on the page.

4.3 Picking up ripe fruits (Selecting the right clues)

"All the clues might seem equal, but some are more equal than the others"

The possible clue set being displayed on our tool now needed a refinement of reordering based on their importance. Hence we needed a formulation to rate them according to their relatedness to the target synset word. We decided to experiment with Pointwise Mutual Information (PMI) as a measure of ranking the clues. We calculate the PMI between the target word and each of the possible clue, and this study has led to an interesting outcome. The relatedness of the word with the possible clue is being successfully reflected in the PMI score and important clues are scoring well on this scale.

$$\operatorname{pmi}(x;y) \equiv \log \frac{p(x,y)}{p(x)p(y)} = \log \frac{p(x|y)}{p(x)} = \log \frac{p(y|x)}{p(y)}.$$

We were concerned about the relevance of the clues with high PMI hence we ranked then according to their occurrences in the corpus as well. **We applied log function on the No. of occurrences of the clue and multiplied it with its PMI score to indicate a new basis for ranking clues.** We came up with another criteria for clue ranking in this way, in the same we also applied the square root function to our cause instead of log, and hence finally settled with three different measures of ranking the possible clues.

4.3.1 Study of PMI scores

PMI scores as mentioned above reflect the relatedness of the possible clues with the target word. Some examples supporting our statement are given below.

Main_Word	Clue_Word	Count(y)	Count(x,y)	PMI	LOG10|PMI
पुष्पित	जीवन	4044	2	2.975973562	10.73377462
पुष्पित	जीवन	4044	2	2.975973562	10.73377462
पुष्पित	लता	3830	1	2.33719588	8.37463741
पुष्पित	रूप	8461	1	1.544593313	6.066269287
पुष्पित	विकास	3009	1	2.578452885	8.968947726
पुष्पित	करती	2540	1	2.747896602	9.356131
पुष्पित	प्रकाश	1901	1	3.037680619	9.960500427

Table 4.1: Top results for the word "पुष्पित"

Main_Word	Clue_Word	Count(y)	Count(x,y)	PMI	LOG10|PMI
अनाथ	अनाथों	18	18	7.461055426	9.365657735
अनाथ	अनाथालय	11	11	7.461055426	7.769888544
अनाथ	मां-बाप	75	2	3.836714493	7.194074724
अनाथ	बताती	44	1	3.676865792	6.042754927
अनाथ	मारती	49	1	3.569235128	6.032707222
अनाथ	चलाना	53	1	3.490763512	6.019039291
अनाथ	मैनेजर	58	1	3.400612415	5.996735128
अनाथ	असहाय	59	1	3.383517982	5.991709625

Table 4.2: Top results for the word "अनाथ"

Main_Word	Clue_Word	Count(y)	Count(x,y)	PMI	LOG10|PMI
अपमान	अपमान	233	233	6.746215421	15.97069302
अपमान	जनक	807	18	2.94326351	8.555694804
अपमान	सहन	304	9	3.226412297	8.010773861
अपमान	मरना	58	2	3.378919591	5.958481394
अपमान	समझ	4805	23	1.404297315	5.170192146
अपमान	कहे	725	5	1.769481678	5.061315696

Table 4.3: Top results for the word "अपमान"

Main_Word	Clue_Word	Count(y)	Count(x,y)	PMI	LOG10|PMI
सामंजस्य	सामंजस्य	104	104	7.552862975	15.23437643
सामंजस्य	रंगों	231	5	3.719883177	8.792360441
सामंजस्य	समन्वय	247	5	3.652912551	8.740312731
सामंजस्य	प्रवृत्तियों	66	2	4.056355414	7.380716893
सामंजस्य	लोक-साहित्य	145	2	3.269276413	7.06610943
सामंजस्य	वैमनस्य	24	2	5.067956325	6.994850293
सामंजस्य	साहित्य	1275	6	2.193920987	6.813243969
सामंजस्य	विचारों	434	3	2.57843073	6.800584568

Table 4.4: Top results for the word "सामंजस्य"

4.3.2 False Positives

Many a time's clues with high PMI scores are simply because their no. of occurrences are low in the corpus itself, or because they are crumpled or mixed up form of two different words which do not occur in the corpus elsewhere. These false negatives have high PMI scores but they are never good clues. Hence we have to come up with a mechanism to discard them as well.

Some of the examples are given below:

सामंजस्य	लोक-साहित्य	145	2	3.269276413	7.06610943
सामंजस्य	साहित्य	1275	6	2.193920987	6.813243969
अनाथ	चलाना	53	1	3.490763512	6.019039291
अनाथ	मैनेजर	58	1	3.400612415	5.996735128
पुष्पित	करती	2540	1	2.747896602	9.356131

4.3.3 Other Possible Measures

4.3.3.1 TF-IDF

As suggested to us, we will also try and use Term Frequency - Inverse Document Frequency (TF-IDF) between the clue words and target word to calculated relatedness and semantic similarity in our next stage later.

4.3.3.2 G^2 using words Matrix

We will experiment with G^2 scores to rate our clues later which can be a good measure of calculating relatedness of the possible clues appearing to us.

4.3.4 Upgrading the corpus

Initially the corpus available to us was only the one being used by Hindi Concordancer which consisted of 198000 Hindi Sentences. We tried to increase our chances of getting better clue words hence we added the most recent Hindi News corpora to it. We cleaned the sense tagged corpus and appended the old corpus to it. We came up with a new set of 225055 Hindi sentences consisting of a whole new set of better possible clues. But there are still times, when looking at our possible set of clues, we tend to think we can always have a better set of possible clues and our corpus can still be improved.

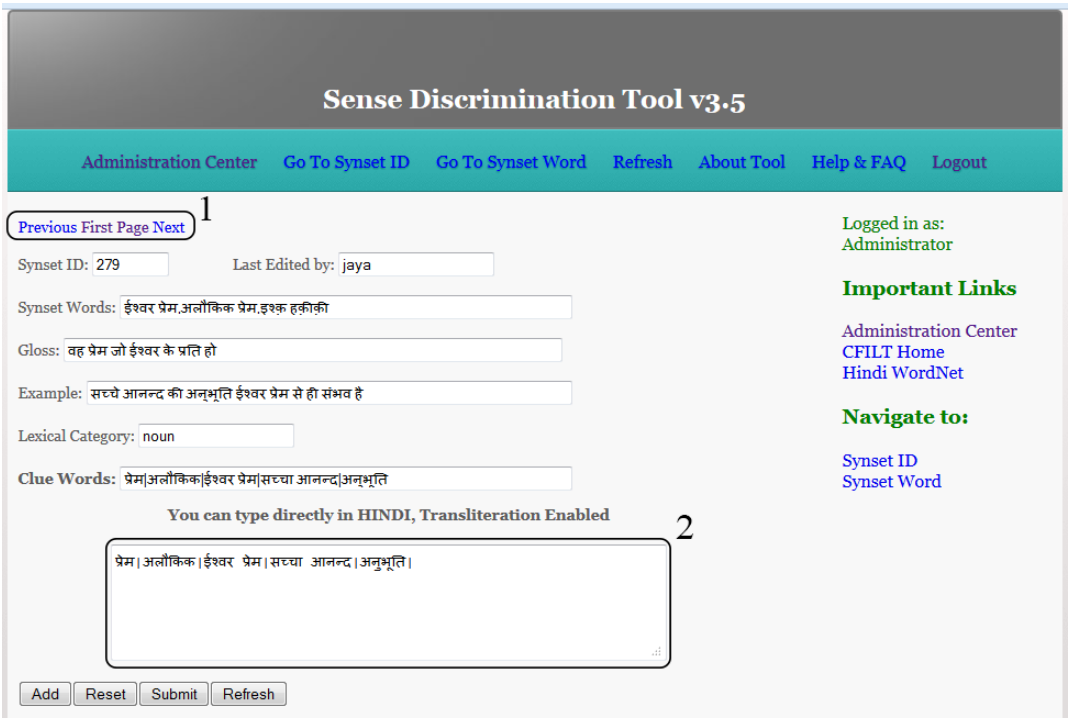
Chapter 5

Demonstration and Documentation

This chapter contains the visual demonstration of the usage of the tool, and the complete documentation of our tool code. It will help you better understand the background working and the workflow of the tool.

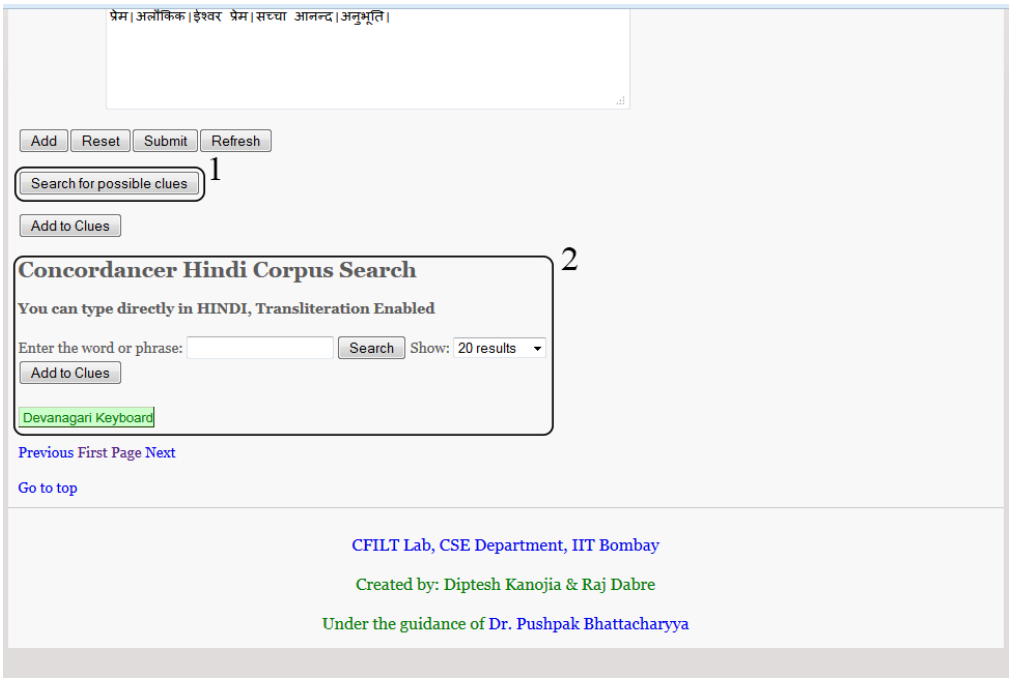
5.1 Demonstration

We will give snapshots of our current tool interface and explain its working below.

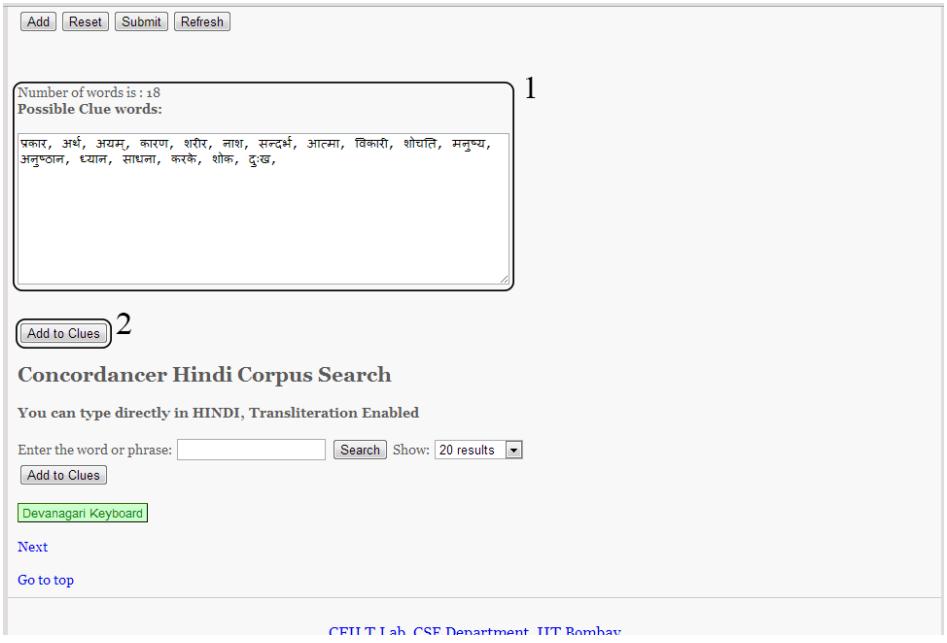


The above screenshot is from the first half of the Tool Home page, which contains two changes from the previous version of the tool.

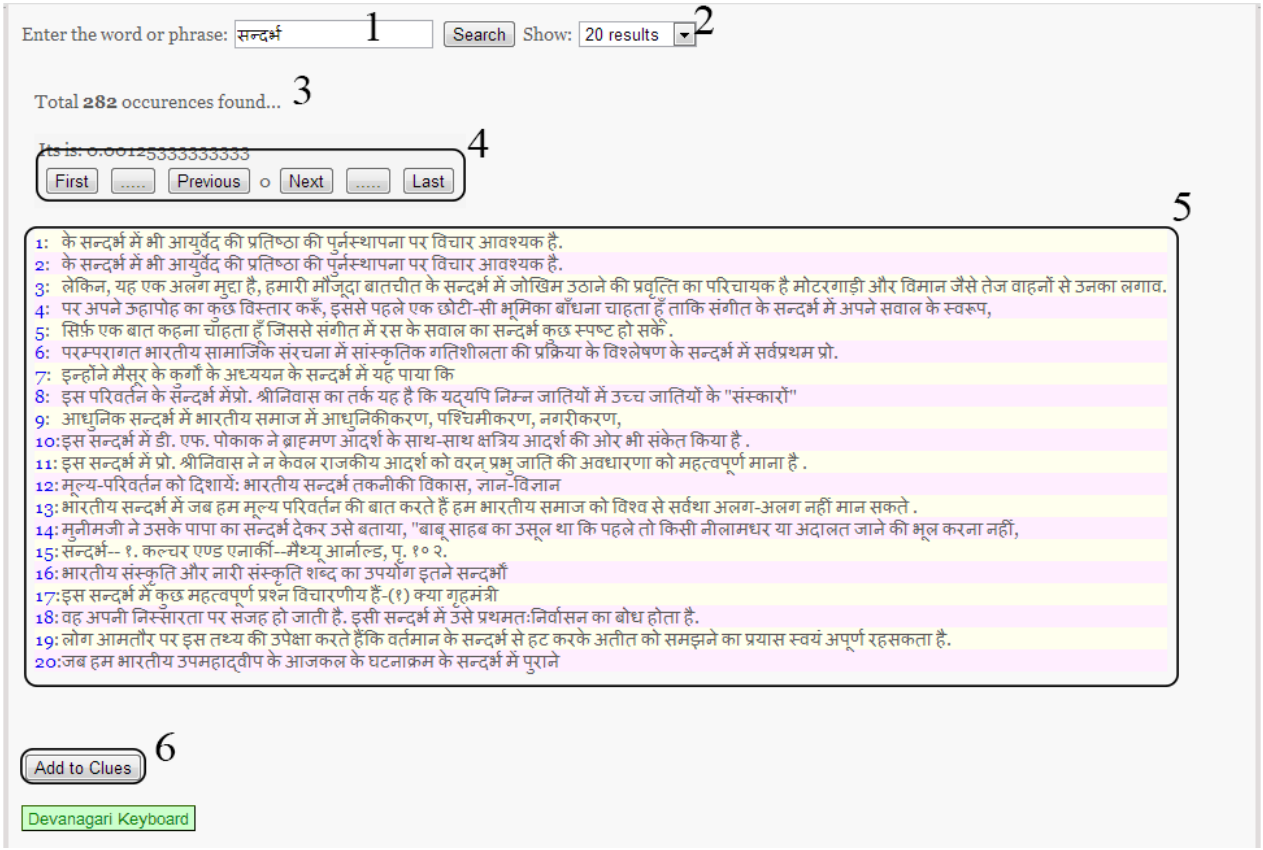
1. The tool now contains navigation hyperlinks on the top as well, for the ease of our lexicographers.
2. The clue box contained on the home page is the same, but is now Transliteration enabled as it says on the page. One can directly start typing a Hindi word and as soon as the word is entered, an API in the background changes it to its Hindi Transliteration.



1. We have added “Search for possible clues” button in the bottom half of the page which on a single click displays results after fossicking the database.
2. This is the concordancer corpus search functionality which enables the user to manually search for a required word in the corpus and detail its results within the page itself.



1. This is the result of the click on the button, the button vanishes and the results of the possible clue search from the database are displayed instead of it.
2. This “Add to clues” button will add any selected text from the “Possible Clue Words” box to the “Clue words” box including the word separator.



1. The search text box where the required text string to be searched across the corpus has to be entered.
2. A drop down selection for user to opt for the number of results at a time on the page.
3. Total number of occurrences of the word found.
4. Navigation box for the results obtained from corpus search.
5. Results of the corpus search.
6. “Add to Clues” button with the functionality of adding anything selected from the corpus results straight to the “Clue Words” box above, saving the time required in copying and pasting related words.

5.2 Documentation

The documentation of this tool contains both black box and white box type documentation. First, we define the logic wise flow and what the tool entails. The tool is an online PHP based interface which works by importing the data from Hindi WN database. We have used MySQL as the back end for data manipulation and storage. The tool starts by displaying a login page where a user must enter his credentials to enter the tool. Unregistered users are required to click on the create login button to go create a login user id and password for them, and their login must be approved by a CFILT administrator or by any of the registered Super Users on the website.

Once the login id is created and approved, a user can log in to the tool and start using it from the home page itself. Home page of the tool is <index.php> contained in the root. It displays extracted data from the database on the first page with the first synset id i.e. <synset id = 1>. The user gets to start clue marking from here itself. Synset words and Synset ID is displayed on the top along with a text box displaying the username of the user who last edited the current clue words, if ever edited. If there is no text field labeled clue words present on the page, there are no entries for the clues of this synset in the database.

User now has to identify the clue words in the *gloss* and *example* fields of the page displayed. Clues can be words or word phrases depending on the user's interpretation of the synset word along with the lexical category it belongs to. Once the user selects clues with mouse selection on the screen, He clicks the add button to put them down in the Clues Text Box given below. User adds as many clues as possible and when the clues are finally complete, He should click Submit button to submit the clues in the database. The clues are added to the database and changes are reflected immediately in most cases. Due to some problems in specific browsers, if the clue changes are not being reflected immediately, the user should click refresh to check the clue changes made in the database. Clicking on refresh will fetch entries of clues from the database immediately.

If there are some clues already added to the database, and user wants to edit them, the clues text box it editable and user can edit the clues present there, when done with the editing, clicking on the submit button is again required to update the clues entry in the database.

It is advised that user only edits the clues if he has a complete idea about the synset word he is presently editing.

The description of the flow of logic in our tool interface is give below:

5.2.1 Tool Home

Link to the tool:

www.cfilt.iitb.ac.in/~diptesh/

which in turn takes you to /index.php

Case 1: If the user is already authenticated [Session variable set from an earlier session in browser]

- /index.php is displayed and the tool can be used from here.

Case 2: If the user is unauthenticated

- User is taken to a page /admin/login.php to enter his login credentials, and then to /index.php.

5.2.1.1 </admin/login.php[1]>

The interface displays simple text boxes in form with action set to execute <login-action.php> and passes the values of username and password entered to the above mentioned file.

Create user button on the file takes user to </admin/register.php> which displays simple text fields which are used to create a user id and password for a user in the tool database.

5.2.1.2 </admin/login-action.php[1]>

It simply calls to include </admin/admin-class.php> and passes $admin variable to class itg_admin(), to be authenticated by function _login_action().

5.2.2 Logged in user interface

5.2.2.1 <index.php>

The tool interface is simple and the data is extracted and displayed on this one page in text fields.

Synset ID: $sid

Synset Words: $words

Gloss: $comb[0]

Example: $comb[1]

Lexical Category: $category

Clue Words: $clues

The data set consists of a field called clue words, which can be manipulated by the user based on his knowledge.

The interface consists of links to next, previous and first synset id which navigate the same page to extracting data of the target synset id and displaying it in the same interface.

After adding the clues to the Clues Text Box, when user clicks on Submit button, a call is made to </addtocsv.php> which adds the clues to the database.

5.2.2.2 <addtocsv.php>

A connection is made to the database 'cluemarker' and table 'tbl_all_synset' which contains column 'clues', which then is updated with a MySQL query replacing it with all what is within Clues Text Box.

5.2.2.3 </admin/admin-class.php[1]>

It calls to include </db/db.php> which makes a connection to the database to extract information about user id in the table 'user' and has all the fuctions related to registration of a user, login of a user, verification, authentication etc. The file contains proper comments in the source code, for further changes.

_login_action() – for login verification.

_register_action – for registration submission.

_check_db() – checks the database for correct username and password, private function.

_authenticate() – for authentication verification of a user, should be used before opening tool interface. Tool opens only if user is authenticated and this is done using this function.

5.2.2.4 <find3.php>

It is called upon the click of the "Search for Possible Clues" button and it searches the corpus for the first synset word. It then takes the first ten sentences, initiates the POS tagger to tag the results and displays the filtered Nouns and Verbs. It passes the final results on to the home page and displays them on the tool. It serves its purpose only in the background.

5.2.2.5 <find4.php>

It is called upon the click of "Search" button in the concordancer below, it searches for the word submitted in the text box and displays the results on the tool home page itself, just below the search button. This file serves its purpose in the background, it searches for the results and passes them onto the home page.

---

[1] A simple version of tool login system was downloaded from www.intechgrity.com and the modified according to our purpose.

5.2.3 Administrator Center

Administration Center is the tool interface for administrator login, which opens up only when an authenticated administrator login in done through tool home. Administrator login redirects to </admin/admin-center.php> which is displayed only for Administrator status users.

This interface has all the user lists i.e. Pending Approval, Registered users

5.2.3.1 </admin/admincenter.php>

This file contains lists of users who are registered to use the tool i.e. approved by the admin, or a super user, and also the list of users whose registration is pending an approval.

Pending List

This list is only displayed if there are any users who have registered for the tool, and have not yet been approved by admin. Also, the users who are banned from using the tool, are brought here, and they cannot login again, until approved by the admin. It displays and option to “Approve” them or “Reject” them.

Registered Users

This list displays the user who are registered as normal users and are approved to view and use the tool. They have no access to administration center whatsoever. The list contains options to “Ban” them, “Delete” them, or promote them to “Super User” status.

Super User

Super users have access to administration center and can go to administration center through a link on index.php i.e. tool interface. Super users can approve or reject members in pending users list. They can also ban users who are creating nuisance in the database. Super users cannot disable other super users, that privilege lies with the admin.

Chapter 6

Conclusions

The development of a tool which allows for annotators to conveniently specify the clues that they use for distinguishing between the various senses of a word is quite crucial in the task of word sense disambiguation. It is further important to utilize these clues so as to build a structure or a framework which allows for reducing the uncertainty of the sense of a particular word.

We went on with further development of the sense discrimination tool to incorporate concordancer corpus search and searching of possible clues using a simple heuristic of collecting adjoining nouns and verbs from the initial search results.

It was indicated that Pointwise mutual information (PMI) score can be used as ranking measure for possible clue words, along with a few other factors such as the relevance of the clue words. Relevance of the clue words depends upon the number of occurrences it has in corpus. We multiplied the occurrences values calculated by their log with individual PMI scores and obtained different results. We imagine that constructing a discrimination net using these values as weights will assist in calculating a score which will reduce uncertainty about the sense of a word.

The discrimination net can help us disambiguate the senses of all the words present in a sentence or corpus text fed to it. We hope that this framework will bring about a newer understanding of WSD. We believe that these clues will give us an in depth analysis of how humans disambiguate and using them, if we can bring down the noise in word senses using these clues, we will easily be able to reach the target sense. Bringing down the noise or the entropy as we have thought of it, will depend on positive clues, which will be available in our database. Moreover, at some point of time, this clue acquisition mechanism will be self-sufficient.

Chapter 7

Future work

We plan to automatically populate the clue words from the already provided data set, and hence facilitate the easy addition of clue words to the whole database. It will reduce a lot of cognitive load and be helpful in generating connections or interconnections between words. However, for now we will keep working to obtain clues manually, since these will be used to validate the quality of the automatically added clues. Overcoming the limitations of a corpus with high coverage is quite crucial which will be remedied by crawling the web for good quality Monolingual corpus.

The greatest challenge will be the formulation of a solid theory which will, in the future, be known as WSD by discrimination net, which will use all the clues obtained automatically and manually. Once the said system is developed, we have to observe its efficacy and accuracy in all domains, for polysemous words as well. It has to be generalized in such a way that it can be projected onto other languages with the same or higher accuracy.

References

Bibliography

Agirre, & Rigau. (1996). Word sense disambiguation using conceptual density. *16th International Conference on Computational Linguistics.* Copenhagen, Denmark.

Arindam Chatterjee, S. J., D.K., and Pushpak Bhattacharyya (2012). A Study of the Sense Annotation Process: Man v/s Machine. *International Conference on Global Wordnets (GWC 2011)*, (p. 8). Matsue, Japan.

Google. (n.d.). Retrieved from http://www.google.co.in

Lesk, M. (1986). Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone. *5th annual international conference on Systems.* Toronto, Ontario, Canada.

Mitesh Khapra, S. S., and Pushpak Bhattacharyya (2009). Projecting Parameters for Multilingual Word Sense Disambiguation. *Empirical Methods in Natural Language Prfocessing (EMNLP09).* Singapore.

Navigli, R. (2009). Word Sense Disambiguation: A Survey. *ACM Computing Surveys*, 69.

Navigli, R., & Velardi, P. (2005). Structural Semantic Interconnections: A Knowledge-Based Approach to Word Sense Disambiguation. *IEEE Transactions On Pattern Analysis and Machine Intelligence.*

Nicholas Wade, B. W. (2005). *The moving tablet of the eye: the origins of modern eye movement research.* New York: Oxford University Press.

Rada, M. (2005). Large vocabulary unsupervised word sense disambiguation with graph-based algorithms. *Joint Human Language Technology and Empirical Methods in Natural Language Processing Conference (HLT/EMNLP)*, (pp. 411-418). Vancouver, Canada.

Salil Joshi, Diptesh Kanojia, and Pushpak Bhattacharyya (2013). More than meets the eye: Study of Human Cognition in Sense Annotation. *NAACL HLT*, (p. 4). Atlanta.

Wikipedia. (n.d.). Retrieved from http://en.wikipedia.org/wiki/Wiki

InTechgrity. (n.d.). Retrieved from http://www.intechgrity.com/