Optimized Word Sense Disambiguation in Hindi using Genetic Algorithm

Sabnam Kumari¹
¹M.Tech Scholar, Department of Computer Science and Engineering, PDM College of Engineering, Bahadurgarh, Haryana
Shabnam022@gmail.com

Prof. (Dr.) Paramjit Singh²
²Professor of Computer Sciences, PDM College of Engineering, Bahadurgarh, Haryana
director_engg@pdm.ac.in

Abstract- Word Sense Disambiguation (WSD) is a problem of computationally determining which “sense” of a word is activated by the use of the word in particular context. Genetic Algorithm is used to figure out the appropriate meanings of polysemous nouns in the given context. To our knowledge this is the first attempt of using Genetic Algorithm for Hindi language. The work is to devise a Genetic Algorithm incorporating Elitism within it. Wordnet for Hindi developed at IIT Bombay, a lexical knowledge base for Hindi, is used. The main focus is on removing the ambiguity of the sense using the context by applying Genetic Algorithm. The work is implemented on Netbeans 7.3. The obtained results show the optimized disambiguation of the Hindi words.

Keywords: Elitism, Genetic Algorithm, Hindi Wordnet, Sense Disambiguation, Word Sense Disambiguation.

I. INTRODUCTION

Word Sense Disambiguation (WSD) is the task of finding the appropriate sense of a word used in a given sentence, when the word may have more than one sense [1]. E.g.

- eagxkbZ ls gj oxZ ds yksx ijs’kku gSaA
  Here oxZ is interpreted as ‘class’.

- lkr dk oxZ mupkl gksrk gSA
  Here oxZ is interpreted as ‘square of number’.

Some words may not be easy to disambiguate as they may have multiple senses that are close to each other. In some senses disambiguation may be impossible altogether using the given sentence. Now consider ^g rks oxZ gSa**

Given only the above sentence, one may translate it as “This is a square shaped figure” or “This is the square of the number”. All are having valid senses of the word oxZ.

In another example lksuk lksuk pkgrh gSA

This sentence can be interpreted as “Sona wants gold” or “Sona wants to sleep” or “Gold wants to sleep” or “Sleep wants Gold” etc. Thus these scenarios require a look at context of the discourse to disambiguate between the possible senses.

Sense Disambiguation [2] is an ‘intermediate task’ which is not an end itself, but rather is necessary at one level or another to accomplish most NLP tasks. Sense Disambiguation involves Sense Knowledge. Sense Knowledge can be represented by a vector, called a sense knowledge vector (sense ID, features), where features can be either symbolic or empirical. The word to be sense tagged always appears in a context. Context can be represented by a vector, called a context vector (word, features). Thus, we can disambiguate word sense by matching a sense knowledge vector and a context vector [1].

II. APPROACHES TO WSD

As in all natural language processing, there are two main approaches to WSD – deep approaches and shallow approaches [3].

[A] Deep Approaches

Deep approaches presume access to a comprehensive body of world knowledge. E.g. consider the word “bass” with two distinct senses: ‘a type of fish’ and ‘tones of low frequency’. Knowledge such as “you can go fishing for a type of fish, but not for low frequency sounds” and “songs have low frequency sounds as parts, but not types of fish” is used to determine in which sense the word is used [1]. These approaches are not very successful in practice, mainly because we don’t have access to such a body of knowledge, except in very limited domains. However, if such knowledge did exist, then deep approaches would be much more accurate than the shallow approaches [3].

There are two types of Deep approach of Word Sense Disambiguation are:

- Selectional restriction- based approaches
• Approaches based on general reasoning with 'world knowledge'

[B] Shallow Approaches
Shallow approaches don’t try to understand the text. They just consider the surrounding words, using information like “if ‘bass’ has words ‘sea’ or ‘fishing’ nearby, it probably is in the fish sense; if ‘bass’ has the words ‘music’ or ‘song’ nearby, it is probably in the music sense.” These rules can be automatically derived by the computer, using a training corpus of words tagged with their word senses. This approach, while theoretically not as powerful as deep approaches, gives superior results in practice, due to our limited world knowledge. The different types of Shallow approaches of WSD are:
• Dictionary-based approaches.
• Machine learning approaches
• Supervised methods
• Semi-supervised
• Unsupervised methods
• Hybrid approach

III. LITERATURE REVIEW
Manish Sinha, Mahesh Kumar Reddy R, Pushpak Bhattacharyya, Prabhakar Pandey and Laxmi Kashyap [3] worked on “Hindi Word Sense Disambiguation” that was the first attempt for an Indian language at automatic WSD. The approach is to compare the context of the word in a sentence with the contexts constructed from the Wordnet and chooses the winner. The output consisted of a particular synset number designating the sense of the word. The evaluation was done on the Hindi corpora provided by the Central Institute of Indian Languages.
Rohan Sharma [6], “Word Sense Disambiguation for Hindi Language” made an attempt to resolve the ambiguity by making the comparisons between the different senses of the word in the sentence with the words present in the synset form of the Wordnet and the information related to these words in the form of parts-of-speech.
Neetu Mishra and Tanveer J. Siddiqui [7], “An Investigation to Semi-Supervised approach for Hindi WSD”, investigated Yarowsky algorithm. After elimination of both, stemming and stop words, the maximum observed precision is 61.7 on 605 test instances.
Sabnam Kumari and Paramjit Singh [1], “Genetic Algorithm based Hindi Word Sense Disambiguation” made an attempt to disambiguate the Hindi words by applying the novel approach i.e. Genetic Algorithms.
Sandeep Kumar Vishwakarma and Chanchal Kumar Vishwakarma [8], “A Graph Based approach to Word Sense Disambiguation for Hindi Language” combined Lesk semantic similarity measures and Indegree algorithms for graph centrality and 65.17% accuracy has been obtained.

IV. HINDI WORDNET AND APIs [4]
Wordnet is a freely available semantic lexicon for the English and Hindi language whose design is inspired by current psycholinguistic theories of human lexical memory. Wordnet for Hindi is produced by people researching in the Centre for Indian Language Technology (CFLIT), IIT-B, under the direction of Prof. Pushpak Bhattacharya [4]. Its design is inspired by the famous English Wordnet. In Wordnet, each part of speech word (nouns/verbs...) is organized into taxonomies where each node is a set of synonyms called synsets. Each synset represents a specific meaning. It includes the word, its definition (gloss), its explanation, and its synonyms. Each Entry in Hindi Wordnet consists of {synsets, gloss, and ontology}.
Current Status of Hindi Wordnet is still under construction. In the version 1.0 is an attempt to cover all the common concepts in Hindi. The present status is as follows:
Total unique words: 95891
Total Synsets: 38070
Linked Synsets: 24582
Last Updated: 9th June 2013
When we install WordNet on our computer a WordNet browser in also installed for external uses.

Java Hindi Wordnet Library (JHWNL)
The Hindi Wordnet data can be accessed by using APIs (Application Programming Interface) written in Java called JHWNL or Java Hindi WordNet Library. These APIs allow searching of synsets containing a particular word and accessing the relations of the synsets. The most important functions in the API are described in Table 1.

<table>
<thead>
<tr>
<th>Class : Dictionary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return Type</td>
</tr>
<tr>
<td>Synset</td>
</tr>
<tr>
<td>IndexWord</td>
</tr>
<tr>
<td>IndexWord</td>
</tr>
<tr>
<td>IndexW</td>
</tr>
</tbody>
</table>
ordSet indexWords (String lemma) with each element in the set corresponding to all POS of the lemma in which synsets are present.

<table>
<thead>
<tr>
<th>IndexWordSet</th>
<th>lookupMorphedIndexWord(POS pos, String lemma)</th>
<th>Returns a set of IndexWord for all root forms of the lemma for the specified POS.</th>
</tr>
</thead>
<tbody>
<tr>
<td>IndexWordSet</td>
<td>lookupAllMorphedIndexWords(String lemma)</td>
<td>Returns a set of IndexWord for all root forms of the lemma for all POS in which synsets are present.</td>
</tr>
</tbody>
</table>

Table 1: APIs of Hindi Wordnet

V. GENETIC ALGORITHM & PARAMETERS

Genetic Algorithm (GA) is a heuristic search algorithm used to find approximate solutions to optimization and search problems using techniques inspired by evolutionary biology. This observation was first mathematically formulated by John Holland in 1975 in his paper, "Adaptation in Natural and Artificial Systems" [9]. It is based on the Darwin’s principle ‘Survival of the Fittest’. In nature, competition among individuals for scanty resources results in the fittest individuals dominating over the weaker ones. In GA, a population of chromosomes (abstract representations of candidate solutions to an optimization problem) evolves toward better set of solutions. Base for evaluating fitness function is WU-Palmer similarity to find relatedness of the words [1]. First we need to decide and set the GA parameters like,

- **Chromosome length**: The length of chromosome is the number of nouns to be disambiguated together.

- **Population size**: 30.

- **Cross-over probability**: In the starting it was taken 0.6 but it was observed that the algorithm is facing the problem of local maxima. This can be solved by introducing more diversity in the population. So we decided it to be 0.8.

- **Mutation Probability**: Mutation is used to get the diversity in population. Generally it is taken as small as 0.02. Due to the problem of local maxima we take it 0.03 Then we generate the initial population. Generally it is a random population. After the initial population is generated, we calculate the fitness value of each chromosome. Then we repeat the steps- Selection-> Crossover-> Mutation-> Evaluation until termination criteria is satisfied. The power of GA is elitism where best individuals at each generation are assured to be at the next generation. When we apply genetic operators on the population there is a chance of losing the best chromosome. To avoid this we preserve the best fit chromosome at each generation. The steps of the Genetic Algorithm using Elitism applied in the work are given in the figure 1 and figure 2 shows the execution of GA.

<table>
<thead>
<tr>
<th>Table 1: APIs of Hindi Wordnet</th>
</tr>
</thead>
<tbody>
<tr>
<td>ordSet indexWords (String lemma) with each element in the set corresponding to all POS of the lemma in which synsets are present.</td>
</tr>
<tr>
<td>IndexWordSet lookupMorphedIndexWord(POS pos, String lemma) Returns a set of IndexWord for all root forms of the lemma for the specified POS.</td>
</tr>
<tr>
<td>IndexWordSet lookupAllMorphedIndexWords(String lemma) Returns a set of IndexWord for all root forms of the lemma for all POS in which synsets are present.</td>
</tr>
</tbody>
</table>

Figure 2: Execution of Genetic Algorithm

VI. DISAMBIGUATOR

Disambiguator disambiguates the words and works in the following manner:
1. Input the sentence.
2. Perform the POS tagging on the sentence
3. Fetch the Nouns from the sentence
4. For Each Noun n fetched from sentence
   If n is available in WordNet
       Add n to List of Nouns N Else
       (Considering n to be inflectional) find all the base forms
       of n (if base form(s) found it means n is inflectional noun)
5. For Each base form b of n
   If b is available in WordNet
       Add b to List of Nouns N End If
   End If
End For
6. Find the number of available senses in WordNet for each
   Noun in N. This number will act as a boundary value of
   an allele in the candidate chromosome in GA.
7. Set the parameters of GA and run it
8. Output of GA is a chromosome that represents the most
   appropriate sense of each Noun in the form of a sense
   number.
9. Find the definitions of each sense for manual verification
   (optional).
This Disambiguator takes the input from a text file and
gives the correct senses along with their definitions corresponding
to each Noun. Results for a particular example are shown in figure 4.

The output consists of the value for best chromosome and
disambiguated meaning of the words. In Figure 3 we can
see that Initial Population is generated and then in Figure
4 the algorithm is applied on 20 generations for the
optimized result for this problem

VII. RESULTS
To evaluate the above output we will find the Recall,
Precision. Recall is defined as the ratio between correctly
disambiguated Nouns and total number of Nouns;
Precision is defined as the ratio between correctly
disambiguated Nouns and total number of answered
Nouns.
No. of total Nouns in the input = 12
No. of correctly disambiguated Nouns = 11
Recall = (11/12)*100=91.6%

VIII. CONCLUSION AND FUTURE WORK
In this work we have chosen the Hindi Wordnet for
fundamental task, viz. disambiguation of Hindi words. To
our knowledge, by using Genetic Algorithm, no attempt
has been made in the past to address the problem of word
sense disambiguation on Hindi language. People have
worked on Hindi language by applying the context based
approach to Word Sense Disambiguation but their success
rates are not that much good around 65%. The algorithm
i.e. Genetic Algorithm gives the optimized results after
disambiguation. The power of GA is ‘Elitism’ where best
individuals at each generation are assured to be at the next
generation. For this purpose a Genetic Algorithm is
designed by incorporating the Elitism in it. Finally the
algorithm is tested on various sets of nouns and obtains
good results. Our system will currently deal with the
nouns only.
In future, words of other parts of speech can be included. With the enrichment of the algorithm the system performance is expected to be very impressive. We can improve the \textit{Recall} by introducing some change in the Genetic Parameters. Genetic Algorithm has been applied to disambiguate nouns in a given sentence. The results are satisfactory and accuracy can be improved further.

\textbf{ACKNOWLEDGMENT}

\textbf{Prof. (Dr.) Paramjit Singh} is the professor in Department of Computer Science and Engineering and Director at PDM College of Engineering, Bahadurgarh, Haryana. I am especially grateful for his guidance and contributions by generously giving his time and carefully reviewing this manuscript.

\textbf{REFERENCES}


