International Journal of Computing and Digital Systems
ISSN (2210-142X)
Int. J. Com. Dig. Sys. 10, No.1 (Oct.-2021)

http://dx.doi.org/10.12785/ijcds/100185

Word Sense Disambiguation in Hindi Language Using Score Based Modified Lesk Algorithm

Praffullit Tripathi[1], Prasenjit Mukherjee[2], Manik Hendre[3], Manish Godse[4], Baisakhi Chakraborty[5]

[1,5]Department of Computer Science and Engineering, NIT, Durgapur, India
[2,3,4]Department of Analytics and IT, PIBM, Pune, India

Received 12 Mar. 2020, Revised 13 Mar. 2021, Accepted 10 Aug. 2021, Published 28 Oct. 2021

Abstract: Hindi is the widely used spoken language in the Indian subcontinent, and is used by more than 260 million Indians citizens. Indian governments has many digital initiatives to serve Indian citizen better, hence Hindi language becomes one of the important languages to serve Indian citizen. The Government initiatives are like smart city, Hospital Services, Common Service Centers, Digital Payment Ecosystem, Pensioners Scheme, Digital Locker and many more. These all initiative are served using mobile and web based applications, which citizens can access easily instead of visiting various government departments. To serve the large Hindi speaking population, it is necessary to handle the ambiguous words which have multiple connotations in any natural language processing task. In this paper, word sense disambiguation for Hindi language is proposed. Proposed method makes use of Lesk algorithm to disambiguate the Hindi words. Novel scoring method is used to assign a sense score to each token of the Hindi sentence. The sense score is calculated based on the gloss, hypernym, hyponym and synonym of the combinations of different sense of tokens. Hindi WordNet database created by CFILT, IIT Bombay is used in the proposed system. The proposed algorithm takes a natural language (NL) sentence in Hindi (Devanagari script) and process the sentence according to the score based approach modeled on the basic Lesk algorithm with the help of Hindi WordNet designed by CFILT IIT Bombay. The solution provided in this paper can be used vividly in various web based applications like Query-Response Systems, Question-Answer Systems, Sentiment analysis, Recommendation systems etc.

Keywords: NLP, Lesk Algorithm, Word Sense Disambiguation, Multi Word WSD, Hindi WordNet.

1. Introduction

Hindi is a Devanagari scripting language. Most of the citizens in the India and its subcontinent uses Hindi language for communication. After Digital India initiative, demand of regional language based web applications are increasing exponentially. Citizens are using application in Healthcare, Transportation, Education, Tourism, Financial and Pension Applications which are in regional languages. Citizens are flexible with their native language and Hindi is the most acceptable language in India and its subcontinent. Native speaker of Hindi is almost 260 million people according to Ethnologue as in [1]. Many Hindi based applications have been developed that are related to the Query-Response Systems, Question-Answer Systems, Sentiment analysis systems, etc. These systems are natural language processing (NLP) based systems, hence has the problem of Word Sense Disambiguation. Understanding of a sense in a Sentence or Word by the NLP based systems is an open research problem. Human languages contain many ambiguous words that create problems at the time of processing sentences. Human languages contain vocabulary that is comprised of words. All the languages are full of words that have multiple meanings associated with it. The meaning of a context is dissected depending on the words used in the sentence. Sometimes same word can have multiple senses [2]. These words are called polysemous words. A sentence with one or more polysemous words in it is said to be ambiguous. Human neural networks (human brain) are way too efficient in disambiguating the correct sense of a word used in a sentence. However, computer systems are not too proficient in it. Human languages are too complex for machines to interpret and understand. It is preferred not to have such ambiguities in computer related applications. Identification of polysemous words and assigning their correct sense is classical problem in NLP. Process of assigning senses to the polysemous words is known as word-sense disambiguation problem.

E-mail: allae.erraissi-etu@etu.univh2c.ma, mouad.banane-etu@etu.univh2c.ma

Word sense disambiguation is one of the major area of interest for NLP researchers. Computer programs do not have the human experience in languages, so automatically correct sense detection of a polysemous word is a difficult task. This is the problem that Word Sense Disambiguation (WSD) tries to solve. WSD is a core research problem in NLP which tries to identify the sense of a word which is used in a given context as in [3] [4]. Most of the WSD algorithms [5] [6] are divided as supervised, unsupervised, and knowledge-based techniques and that can be utilized to improve many NLP based applications like Information Retrieval [7] [8], Sentiment Analysis [9] [10], Machine Translation [11] [12] or Text Summarization [13] [14]. Supervised learning [15] requires large volumes of human tagged data in order to find set of rules to disambiguate future queries. There are two drawbacks to this approach. First, human interaction is required. Second, no set of rules can always dissect correct meaning of the word. On the other hand, unsupervised learning [16] uses classical sources like dictionary based WordNet [17] etc. to disambiguate senses. Lesk algorithm falls under the category of knowledge base based algorithm. Classical Lesk algorithm offers a score based approach to find overlap between different senses of target word and words nearby to it. The Lesk algorithm is one of the most well-known algorithm for word sense disambiguation which is introduced by Michael E. Lesk [2] in 1986. A simplified version of the Lesk algorithm is to compare the dictionary definition of an ambiguous word with the words contained in its neighborhood. Versions have been adapted to use WordNet. The proposed work presents an adaptation of Lesk algorithm that includes dictionary to disambiguate ambiguous Hindi words from a Hindi sentence. Rather than using a standard dictionary, this algorithm employs a different kind of dictionary called as WordNet. Traditional dictionary arranges words alphabetically while WordNet organize the words semantically. Glosses refer to the definitive meaning of the word. Overlapping between the glosses of the neighbouring words plays a key role in Lesk's approach [2]. The gloss of a word provides us the following information.

1. List of different senses of the word in WordNet. Each sense belongs to a different synset.
2. Information about the context in which word appears.

A scoring method is provided with the algorithm in order to give sense score to each set of combination. This scoring method is carried out by considering all pairwise matching such that each element of the pair belong to the same detail of separated words. For a given query having 'n' tokens each having 'k' senses, there will be total 'n^k' different combinations. Proposed scoring method is applied to all these different combinations to give sense scoring and choose appropriate sense. Before applying word sense-count firstly, we have to trim the query so as to reduce the useless words (stop words) for unnecessary computation. Further, we develop the query by searching stemmed words for validity. If there is no such validity in WordNet, we reject the word just like stops words. After a data set generation, we implement the Lesk algorithm for further processing of sentences. Word sense disambiguation is an open problem in computational linguistic where it detects correct sense of a word that is used in a sentence. In the Simplified Lesk algorithm, the meaning of a word is calculated by finding the overlaps between the context and the dictionary meanings of the word. The meaning for which maximum overlap takes place is taken as the word sense. The word sense of each word is found individually. Internet is emerging as a major source of communication in India. It is expected to have large data analysis requirement in regional local languages with the availability of internet services is rural areas. Hindi language contains many ambiguous words handling which is difficult for NLP systems because meaning of a same word is different in different sentences with different contexts.

In this paper, we have proposed to modify the Lesk Algorithm to disambiguate ambiguous Hindi words efficiently. The score and combination based approach have been applied on Lesk algorithm. NLP based steps like tokenization, lemmatization, POS tagging are used to process the Hindi sentences. The Hindi WordNet is used to extract the correct sense of an ambiguous Hindi word. The International Phonetic Alphabet (IPA) notation [1] for Hindi words is used in this research work. The proposed algorithm takes Hindi sentences as an input. The output will be generated as correct sense of the Hindi word. Many WSD systems have been discussed briefly in Related Works Section (Section 2). The Architecture of this WSD system has been elaborated in Section 3. The Time Complexity of Proposed system has been calculated in Section 4 where Methodology has been given in Section 5. Applications of the system have been stated in Section 6. Finally, the Future Work and Conclusion has been drawn in Section 7 and Section 8 respectively.

Related Works

Word sense disambiguation is one of the key problems in the NLP domain. A large volume of research is going on for the selection of an efficient WSD algorithm. Majority of the unsupervised learning solutions use Lesk Algorithm as their underlying algorithm. Basic Lesk

algorithm is implemented for the English corpus. Some of the languages does not have rich corpus like English. Such languages require some variations in classical Lesk algorithm in order to disambiguate senses efficiently. Rajat Pandit proposed dependency tree based lesk variant to perform WSD for low resource language like Bengali [18]. Shancheng Tang proposed a solution using deep Chinese word sense disambiguation method which is based on sequence to sequence that improved performance by 11.48% [19]. Evaluation of different WSD algorithm is a major sub problem in the NLP domain. Alessandro Raganato designed a unified evaluation system and measure the performance of various word sense disambiguation algorithms. This paper offers two main contributions. The first contribution is (1) standardizing the WSD datasets and training corpora into a structured format, (2) semi-automatically annotation conversion from any dataset to WordNet 3.0, and (3) pre-processing the datasets by consistently using the same pipeline. Second, the evaluation framework has been used to perform a fair quantitative and qualitative empirical comparison of the main techniques that is discussed in the WSD literature, including the latest advances based on neural networks [20]. Quang-Phuoc Nguyen designed a lexical semantic network for Korean language which is useful for Korean morphological and word sense disambiguation [21]. The Lexical semantic network [21] uses largest Korean LSN that consist of Lexical networks of nouns, predicates, and adverbs. Each node is connected with other node using six types semantic relation that are hyponymy, synonymy, similarity, antonym, part-whole, and association relations. To represent a certain sense of a word, each node is consisted with a word and its sense code. Korean morphological analysis is difficult because different types of morphems and parts of speech are encoded into the same eojeol. The meaning of a morphem can be changed because of different parts of speech tagging with the morphems. The morphological analysis refers to discover the correct set of sense in a given context. In Korean morphological analysis, the conventional methods are a) Segment the input eojeol into morphemes and b) tag POS to each morpheme as in [21]. The application of WSD is not limited to any particular domain. Saeed Seifollahi, Mehdi Shajari use WSD to analyze the sentiments of news headlines in order to predict their impact on the stock prices. Some of the applications require heavy computations. To make the application time efficient, several heuristics can be utilized to improve the performance. Genetic algorithms are used widely to address various hard optimization problems. Zankhana B. Vaishnav applied Knowledge-Based Approach to disambiguate polysemous words Using Genetic Algorithm. Genetic approach proposed used Indo-Aryan WordNet for Gujarati language as lexical database as in [23]. S. N. Mohan Raj propose method to eliminate ambiguity due to homonymy using cluster and deep learning approach. Homonymy ambiguity is a problem in Malayalam language. This problem is attempted by the Authors in [24]. The system uses POS tagging lemmatization and sense annotation. The neural network has been used in deep learning method and this method is using the corpus to disambiguate the homonymous words in Malayalam as in [24]. Jagbir Singh proposed a word Sense disambiguation system that is based on Punjabi language. The enhanced Lesk approach has been utilized to extract correct sense of ambiguous Punjabi words. The methodology of this system is supervised learning and Indo WordNet has been used for disambiguating Punjabi words as in [25]. Lekshmi R Pillai proposed a question answering system to predict an answer for an input question. This model is based on a combined approach using word sense disambiguation (WSD) and semantic role labelling (SRL). The proposed system is a factoid sense based question generation system. The Lesk tool has been used for WSD where the senna tool has been used for SRL. They used Lesk tool for WSD and senna tool for SRL which are based on the sense affiliated with the sentence system generates questions that are semantically solvable as in [26]. Word sense disambiguation can be solved using three kinds of approaches, knowledge based, corpus based and hybrid approach. Himdweep Walia proposed a Naïve Bayes based WSD approach that has shown higher accuracy than other implementations as in [27]. Training data is an issue to disambiguate ambiguous words in Persian language that has been faced by the Author [28]. The Machine Learning (ML) algorithm with minimum supervision has been considered to solve this problem. Various news articles have been used as main source of the reference corpus. This method uses some predefined features of target words to disambiguate senses of the word as in [28]. G. Sajini designs a user interface where user manually enters a sentence in Hindi with polysemy word. This system [29] identifies the polysemous words and list one or more meanings that are associated with that word. Using machine learning techniques, this system identifies the correct meaning of the polysemy word which is based on the given context as in [29]. Author [30] proposed a WSD system which is based on Naive Bayes classifier (ML technique). Sense corpus and Ambiguous corpus have been used by this WSD system.

TABLE I. COMPARATIVE STUDY AMONG SIMILAR TYPE WSD SYSTEMS WITH THE PROPOSED WSD SYSTEMS

Sl. No.	Authors	WSD based Systems	Technique(s) used in similar type WSD System	Technique(s) used in proposed WSD System
1	Rajat Pandit et al. [18]	Word Sense Disambiguation by Improvised Lesk Algorithm [18]	The LESK algorithm has been improved for Word sense disambiguation (WSD) on Bengali language. The dependency tree based Lesk has been used to perform the WSD.	The classical Lesk algorithm has been modified for the WSD in Hindi language. The proposed algorithm uses score based approach.
2	Shancheng Tang et al. [19]	Deep Chinese Word Sense Disambiguation Method [19]	Chinese WSD method has been used in this system. The Deep learning method has been used for feature extraction of Chinese text.	The Hindi sentence is used to process according to the score which is modeled on basic Lesk algorithm. The score is assigned on each combination of sense. The largest sense score is the output of this system.
3	Alessandro Raganato et al. [20]	A Unified Evaluation Framework on Word Sense Disambiguation [20]	An evaluation framework has been designed, that is based on various WSD. The first step of evaluation is standardizing the WSD datasets and training corpora into a unified format, The second step is Converting annotation from any dataset to WordNet and third step is pre-processing of dataset using same pipeline. The proposed frame work performs a comparison which based on quantitative and qualitative.	The proposed system is using supervised method of Lesk algorithm. The Hindi WordNet of CFILT, IIT, Bombay has been used to implement this system. This system is able disambiguate ambiguous Hindi words easily.
4	Quang-Phuoc Nguyen et al.[21]	Korean-Vietnamese Neural Machine Translation System [21]	Korean is a morphologically rich language where Vietnamese is an analytical language. Korean language contains word ambiguities which is a problem for Neural MT. The Korean knowledge base has been prepared for the lexical semantic network which is used for morphological analysis and WSD in Korean language. Another large Korean-Vietnamese corpus has been prepared for the Vietnamese word segmentation method.	Hindi is an Indo-Aryan language. Hindi is used by most of the people in India and Indian subcontinent. Understanding a Hindi Sentence is a real issue because Hindi sentence contains many ambiguous Hindi words. This problem has been solved by the Lesk algorithm using Hindi WordNet.The Classical Lesk algorithm has been modified.
5	Saeed Seifollahi et al. [22]	Word Sense Disambiguation in Sentiment Analysis [22]	A sentiment analysis has been done on the News headlines. The WSD has been used on sentiment analysis to predict the impact of the stock price. They have used the news headlines as input of the proposed system.	The Score based approach has been applied on Lesk algorithm to disambiguate the ambiguous words. A sense score has been assigned to each possible combination of senses where largest sense score is the output of this proposed system.
6	Zankhana B. Vaishnav et al. [23]	Word Sense Disambiguation in Gujarati Language [23]	A genetic algorithm has been used to Disambiguate polysemous words. The proposed system is based on the Gujarati language. It uses Indo-Aryan WordNet database for the Gujarati as lexical database.	A score based modified Lesk algorithm has been used to disambiguate the ambiguous words. The proposed system is based on Hindi language.
7	S. N. Mohan Raj et al. [24]	Word Sense Disambiguation of Malayalam Nouns [24]	This system uses WSD on Malayalam language. The proposed work concerned about the homonymy ambiguity. To resolve the homonymy ambiguity, two approaches have been given- Clustering and Deep learning. Clustering is supervised method where POS tagging, lemmatization and sense annotation have been used. The Deep learning approach is based on neural network that uses corpus for disambiguating homonymous words.	The proposed system uses Hindi words for disambiguation. The Hindi language is Devanagari Script and this script is containing many ambiguous words. To extract the correct sense, The Lesk algorithm has been applied after modification. Stop words elimination, Stemming, POS tagging, Sense generation are most popular steps have been applied on this proposed system.

8	Jagbir Singh et al. [25]	Word Sense Disambiguation in Punjabi [25]	This work is analyzing the correct meaning of the ambiguous words in Punjabi language. An enhanced Lesk approach have been utilized to extract the correct sense of the ambiguous words. The supervised learning methodology and Indo WordNet have been utilized to develop this system.	The proposed system is able to extract correct sense of the ambiguous Hindi word. Modified Lesk has been used. Cltk library has been used for Hindi Stop Words detection. Indicnlp library has been used for the Stemming and POS tagging of Hindi words. The combinations of senses have been generated by the proposed algorithm to assign score. The largest score of a sense combination detected as output.
9	Lekshmi R Pillai et al. [26]	Word Sense Disambiguation for Question-Answering System [26]	This is a factoid sense based question-answering system. The WSD and Semantic Role Labelling (SRL) approaches have been applied on this system. The Lesk is used for WSD and Senna is used for SRL.	This is modified Lesk algorithm based Hindi WSD system. Score approach has been modeled on Lesk algorithm and sense combination has been generated to assign score. Output is dependent on to the score.
10	Himdweep Walia et al. [27]	Gurmukhi Word Sense Disambiguation [27]	Three kind of approaches is used in WSD. The first approach is knowledge base, the second approach is corpus based, and third approach is hybrid based. The Naïve Bayes classification algorithm is used on WSD to disambiguate Gurumukhi words. The Punjabi corpus have been used for Gurumukhi words.	Dataset generation is a one of the important part of this proposed WSD system. The Hindi WordNet has been used to filter out various Hindi word senses. Extracted information from the WordNet has been assigned as value. The dataset has been represented from this extracted information. This dataset has been utilized to evaluate sense score.
11	Mohamadreza Mahmood et al. [28]	Persian word sense disambiguation [28]	Persian language based WSD problem has been solved by the Machine Learning (ML) algorithm where minimal supervision has been considered. The corpus has been represented from various news articles. The proposed system uses some predefined features of targeted words and collaborative learning method.	Hindi language based WSD problem has been solved by the modified Lesk algorithm which is score based. Combinations of senses are generated to assign the score for determination of correct sense of a Hindi word. The most popular Hindi WordNet has been utilized which is Developed by the CFILT, IIT, Bombay.
12	Saiba Nazah et al. [32]	Word Sense Disambiguation of Bangla sentences [32]	This WSD system is based on Bengali words. Naïve Bayes classifier and artificial neural network (ANN) have been used to disambiguate the Bengali Words. The Naïve Bayes classifier has been use in training phase and ANN has been used to detect the correct sense of the word.	The Score and Combination based approach have been used to modify Lesk algorithm. The algorithm is flexible to handle Hindi ambiguous words. The natural language processing steps have been utilized. The dataset has been generated using the Hindi WordNet.

The Sense corpus contains synsets, synonyms and antonyms where Ambiguous corpus contains all possible ambiguous words. The experimental result of this system is promising to disambiguate the ambiguous words as in [30]. Unsupervised approach is a most popular approach in Machine Learning. A WSD system has been described by the Author [30] which is unsupervised approach and supervised approach has been bypassed by this system using simulation of the semantic inference process that is performed by human language users as in [31]. Another Bengali WSD system has been proposed by the Author [32] that uses Naïve Bayes classifier and Artificial Neural Network (ANN) to disambiguate the ambiguous Bengali words in Bengali Language. Naïve Bayes classifier has been used for the training phase and ANN has been utilized to predict and detect the correct sense of an ambiguous Bengali word as in [32]. Adapted Lesk method has been used by the Author [33] in their WSD system. The WordNet has been utilized instead of a standard dictionary. The WordNet contains a hierarchy of a semantic relation. The proposed system takes a single target word and its surrounding words as Input. The output is generated as sense of the target word. The comparison has been done on glosses of involved words and related words like hypernym, hyponym, holonym, meronym, troponym, and attribute that are available in WordNet. The sample data from SENSEVAL-2 has been utilized for method evaluation and this system has achieved 32% overall accuracy as in [33]. A comparative study has been done on similar type WSD system and proposed WSD system. The comparative study has been given in Table 1.

2. ARCHITECTURE OF THE PROPOSED SYSTEM

Proposed System architecture (Figure-1) takes a natural language sentence in Hindi as input. It first removes Stop Words from the sentence. Stemming and POS tagging is major step for any natural language application. After eliminating the stop words from the input sentence, refined query is stemmed word by word. Further, refined query is POS tagged and only tokens with multiple senses are left. Each of these tokens are queries in the WordNet semantic database and all the information usable in future is separated in a distinct variable named as dataset. Different possible sense combinations are generated one by one. Each of the combination is assigned a value evaluated using proposed scoring method. The sense combination with maximum sense score is output of the algorithm. Output is returned to the client in the form of word sense pairs.

The proposed algorithm is modularized into following 6 steps. Each of these modules are explained in details in subsequent sections.

1. Post NL Sentence in Hindi
2. Stop Word Elimination
3. Stemming and POS Tagging
4. Dataset Generation
5. Modified Lesk Algorithm Implementation
   a. Sense Combination Generation
   b. Sense Score Assignment
6. Output (Word Sense Pair)

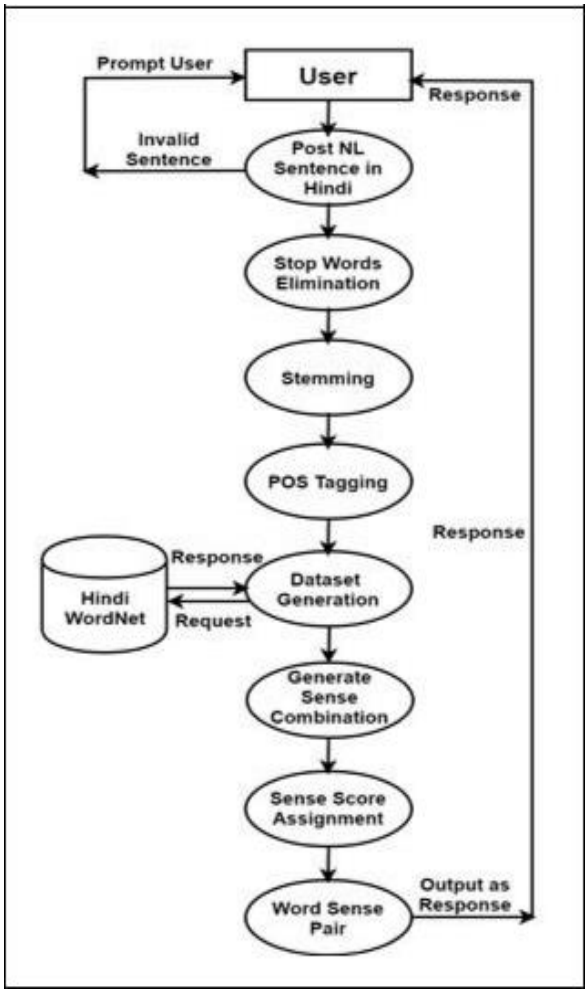

Figure 1. Architecture of the Proposed WSD System

3.1. Algorithm

Basic Algorithmic Steps of this Proposed WSD System has been described here.

i. **Post NL Sentence in Hindi**

The proposed WSD system will read natural language sentence in Hindi. It will check for sentence validation. If sentence is not validated then system will prompt to user. Otherwise the sentence will go for next step.

ii. **Stop Words Elimination**

Each language has a variety of stop words which is of no use while processing them for natural language processing applications. So it would be better to eliminate these stop words prior to advance of further processing.

Few example of Stop Words used in Hindi language are listed here. These are in Devanagari script.

['हैं'(□e□̃), 'है'(□□□), 'हैं'(□□□□̃), 'हह'(□□[6]), 'ही'(□i□[6]), 'हो'(□o□), 'हे'(□e□), 'से'(se□), 'अत'(ə[5]t[2]), 'के'(ke□), 'रहे'(r□ e□), 'का'(ka□), 'की'(ki□[6]), 'हक'(k□[6]), 'तो'(t[2]o□), 'ने'(ne□), 'एक'(e□k), 'नहीीं'(n□i□[6]), 'पे'(pe□), 'मैं'(me□̃), 'वाले'(□[4]a□l e□), 'सकते'(skt[2]e□), 'वह'(□[4]□), 'वे'(□[4]e□), 'कई'(ki□[6]), 'होती'(□o□t[2]i□[6]), 'आप'(a□p), 'यह'(j□), 'और'(□□r), 'एवीं'(e□□[4]̃), 'को'(ko□), 'मे'(me□), 'दो'(d[2]o□), 'थे'(t□[2]e□),'यहद'(jd[2]□[6]), 'उनके'(□[6]nke□), 'थी'(t□[2]i□[6]), 'पर'(pr), 'इस'(□[6]s), 'साथ'(sa□t□[2]), 'हलए'(l□[6]e□), 'जो'(d□ o□), 'होता'(□o□t[2]a□), 'या'(ja□), 'हलये'(l□[6]je□), 'द्वारा'(d[2]a□r a□), 'हुई'(□i□[6]), 'जब'(d□b), 'होते'(□o□t[2]e□), 'व'(□[4]), 'न'(n), 'उनकी'(□[6]nki□[6]), 'आहद'(a□d[2]□[6]), 'सकता'(skt[2]a□e□), 'उनका'(cka□), 'इतना'(□[6]t[2]na□), 'इतयाहद'(□[6]t[2]ja□d[2]□[6]), 'हजस'(d□□[6]s), 'उस'(□[6]s), 'कै 'से(k□□se□), 'हूँ'(□u□[6]̃), 'ना'(na□), 'कहह'(k□□[6]), 'सम'(sm), 'र्'(r), 'कहूँ'(k□□̃), 'बस'(bs), 'अपना'(ə[5]pna□), 'यही'(j□i□[6]), 'कहीीं'(k□i□[6]̃), 'हाूँ'(□a□□̃), 'मैंने'(m□□ne□), 'जहूँ'(d□□□̃), 'सब'(sb), 'यह'(j□), 'था'(t□[2]a□), 'तुम'(t[2]u□m), 'ये'(je□), 'जे'(d□ e□), 'भी'(b□i□[6]), 'हम'(□m), 'अब'(ə[5]b), 'ऐसे'(e□se□), 'वहाूँ'(b□a□□̃), 'क्या'(kja□), 'ओर'(□□o□r), 'इसी'(□[6]si□[6]), 'सके'(ske□), 'कभी'(kb□i□[6]), 'हर'(□r), 'मेरी'(me□ri□[6]), 'कम'(k m), 'सा'(sa□), 'उन्हें'(□[6]n□e□□̃), 'मेरे'(me□re□), 'उन'(□[6]n), 'कु 'छ(k□[6]t□□), 'इन'(□[6] n), 'ऐसा'(e□sa□), 'जहा'(d□□ a□), 'तीन'(t[2] i□[6] n)**]**

Above is the exclusive list of all stop words in Hindi language. Python Cltk library (https://pypi.org/project/cltk/) has been used for the stop words. In code, these stop words are represented by STOPS_LIST list variable. This list contains many Hindi words that are related to the Determiner, Pronoun and Adverb, Preposition, Wh word of English.

Examples:

a) ***Determiner***: 'नहीं'(n□i□[6]) (No),'कुछ'(k□[6]t□□) (Few), 'हर'(□r) (Each), etc.

b) ***Pronoun***: 'उनकी'(□[6]nki□[6]) (him/her), 'हम'(□m)(I), 'अपना'(ə[5]pna□)(My), 'तुम'(t[2]□[6]m)(You), 'मेरी'(me□ri□[6]) (My), 'वह'(□[4]□) (He/She) etc.

c) ***Adverb***: 'यही' (j□ i□[6]) (here), 'अब' (ə[5] □[4]) (Now), etc.

d) ***Preposition***: 'द्वारा' (d[2]a□r a□) (By),'जब' (d□b) (If), etc.

e) ***Wh Words***: 'कैसे' (k□□se□) (How), 'कहँ' (k□□̃) (Where), 'क्या', 'कि' (kja□) (k□[6]) (What), 'जहा' (d□□a□) (Where), etc.

These words are Stop words which are not very important for the Hindi sentence processing. In this step, Stop words are eliminated from the natural language sentence. Example sentence has been given here in Figure 2 and 3.

Natural language Sentence: मतकेआधारपरफैसलाहुआ। (mt[2] ke□ a□d□[2]a□r pr p□□□sla□ □□[6]a□) **(The decision was made according to the openion.)**

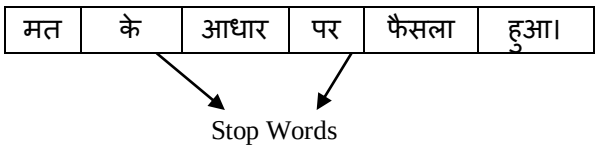

Figure 2. Example sentence with Stop Words

'के' and 'पर' are filtered out as both of them are stop words. Refined query is -

मत	आधार	फैसला	हुआ

Figure 3. Example sentence after Stop Words Elimination

Pseudo code of Stop Words Elimination step has been given here.

Pseudo Code: Stop Words Elimination

```
# install morfessor library
import itertools
import math
from pyiwn import pyiwn
iwn = pyiwn.IndoWordNet('hindi')

# initialize indicnlp library
from indicnlp import common

# place the directory of indic_nlp_resources
common.set_resources_path(r"C:\Users\Praffullit
Tripathi\.spyder-py3\Proj\indic_nlp_resources")
from indicnlp import loader
loader.load()
from indicnlp.morph import unsupervised_morph
analyzer=unsupervised_morph.UnsupervisedMorphAnalyzer('m
r')

# include STOPS_LIST using cltk library
from cltk.stop.classical_hindi.stops import STOPS_LIST
from cltk.tokenize.sentence import TokenizeSentence

matra = ["ॅ","ं","ः","ँ","ा","ि","ी","ु","ू","ृ","ॄ","ॅ",
"े","ै","ॉ","ो","ौ","्"]
karak = ["ने","को","से","के","द्वारा","लिए","में","पर", "का","की",
"के","रा","रे","री"]

print(STOPS_LIST)

STOPS_LIST.append(karak)

def tokenizer(string):
# this tokenize sentence only
tokenizer = TokenizeSentence('hindi')
   hindi_text_tokenize = tokenizer.tokenize(string)
   filtered_token_list = []
for word in hindi_text_tokenize:
if word not in STOPS_LIST:
   filtered_token_list.append(word)
   hindi_text_no_stop_word = ""
for it in filtered_token_list:
   hindi_text_no_stop_word=hindi_text_no_stop_word+it+" "
return hindi_text_no_stop_word
```

iii. Stemming and POS tagging

Filtered query sentence (query without stop words) is again parsed word by word in order to stem the root word from each filtered token. Indicnlp library (https://pypi.org/project/indicnlp/) provides support to stem words using morphological analysis of their WordNet resource. While stemming, we further improvise query by searching stemmed word into WordNet for any valid sense. We search this stem word as either noun, adverb, adjective or verb as all other part of speech is of no use. If stem word have no valid senses as above stated part of speech then we simply reject the word.

After stemming we get tokens as below:
["मत" (mt[2]) (Opinion), "आधार" (a□d□[2]a□r) (according),"फै सला"(p□□□sla□) (Decision)]

The Pseudo Code of Stemming and POS Tagging has been given here.

Pseudo Code: Stemming and POS Tagging

```
def tokenizer_and_stemmer(string):
   hindi_text_no_stop_word=tokenizer(string)
   hindi_text_morphe=""
# tokenize as well as morphological splitting of words
for word in hindi_text_no_stop_word.split():
   if len(iwn.synsets(word, pos=pyiwn.NOUN))>0 or
len(iwn.synsets(word, pos=pyiwn.VERB))>0 or
len(iwn.synsets(word, pos=pyiwn.ADJECTIVE))>0 or
len(iwn.synsets(word, pos=pyiwn.ADVERB))>0:
   hindi_text_morphe = hindi_text_morphe+ word +" "
elif len(iwn.synsets(word+"ना", pos=pyiwn.VERB))>0:
   hindi_text_morphe = hindi_text_morphe + word +"ना"
else:
   morphed_text_temp =
analyzer.morph_analyze_document(word.split(' '))
   temp_str = ""
if morphed_text_temp != word:
for word_iter in morphed_text_temp:
if word_iter in matra:
   word_iter = temp_str + word_iter
   if len(iwn.synsets(word_iter, pos=pyiwn.NOUN))>0 or
len(iwn.synsets(word_iter, pos=pyiwn.VERB))>0 or
len(iwn.synsets(word_iter, pos=pyiwn.ADJECTIVE))>0 or
len(iwn.synsets(word_iter, pos=pyiwn.ADVERB))>0:
   hindi_text_morphe = hindi_text_morphe+ word_iter+" "
elif len(iwn.synsets(word_iter+"ना" ,pos=pyiwn.VERB))>0:
   hindi_text_morphe = hindi_text_morphe+ word_iter+"ना"
   temp_str = word_iter

   final_token_list=[]

for word in hindi_text_morphe.split():
   final_token_list.append(word)
```

```
for word_it in final_token_list:
print(word_it),
return final_token_list
```

iv. Dataset Generation

After filtering and stemming the query sentence, we're querying wordnet to filter out various senses of the tokens. A dictionary of tokens acts as the key and all the extracted information from the wordnet are assigned as the value to the tokens or the key. This algorithm requires hypernym, gloss, hyponym and synset values in further steps to evaluate sense scores.

Structure of the Dataset: DATASET['word'][in_1]
Dataset is a dictionary, each token of the query sentence acts as key. Dataset is formed in a nested structure. Each sense of the token is further classified according to their part of speech. in_1 iterator uses to iterate through highest level of depth which access the meaning in the following order

"मत"(m t[2]), "आधार"(a□d□[2]a□r) and "फैसला"(p□□□sla□) have 5,14 and 2 possible senses. All these senses are encapsulated in variable name dataset.

TABLE II. SIGNIFICANCE OF INDICES USED IN DATASET

Index Value	Represented property
0	Gloss of the word itself
1	Gloss of the hypernym of the word
2	Gloss of hyponym of the word
3	Word synonymous to the word
4	Part of speech of the in1 th sense

Pseudo Code: Dataset Generation

```
def dataset_generation(token_list):
   reject_words = {",ने'}
dataset = {}
for iter in token_list:
if iter in reject_words:
continue
else:
dataset[iter] = []

# iterate for each word 4 times once as noun ,verb, adj and verb
respectively
for pos_iter in range(0,5):
if pos_iter == 0:
syns = iwn.synsets(iter, pos=pyiwn.NOUN)
elif pos_iter == 1:
syns = iwn.synsets(iter+"ना", pos=pyiwn.VERB)
elif pos_iter == 2:
syns = iwn.synsets(iter, pos=pyiwn.VERB)
elif pos_iter == 3:
syns = iwn.synsets(iter, pos=pyiwn.ADJECTIVE)
elif pos_iter == 4:
syns = iwn.synsets(iter, pos=pyiwn.ADVERB)
if len(syns)>0:

# Set Flag 1 To Remove Stop_Words While Formation Of
Dataset
flag=1
for i in range(0,len(syns)):
temp = []
if flag == 1:
    gloss_main = tokenizer(syns[i].gloss())
temp.append(gloss_main)
else:
temp.append(syns[i].gloss())
if pos_iter<=2:
hypernym = syns[i].hypernymy()
else:
hypernym = []
    hypernym_str =""
for k1 in range(0,len(hypernym)):
if flag == 1:
   gloss_hyper = tokenizer(hypernym[k1].gloss())
   hypernym_str = hypernym_str + gloss_hyper +" || "
else:
                hypernym_str = hypernym_str +
hypernym[k1].gloss() +" || "
temp.append(hypernym_str)
if pos_iter<=0:
hyponym = syns[i].hyponymy()
else:
hyponym = []
    hyponym_str =""
for k2 in range(0,len(hyponym)):
if flag == 1:
    gloss_hypo = tokenizer(hyponym[k2].gloss())
    hyponym_str = hyponym_str + gloss_hypo +" || "
else:
    hyponym_str = hyponym_str + hyponym[k2].gloss() +" || "

temp.append(hyponym_str)
    syn_words = ""
for j in syns[i].lemmas():
   syn_words = syn_words+ j.name() +" || "
temp.append(syn_words)
temp.append(syns[0].pos())
dataset[iter].append(temp)
return dataset
```

v. Modified Lesk Algorithm implementation

1. *Sense Combination Generation*

This algorithm generates all the different combinations of sense of the tokens as depicted in the Figure 4 and uses scoring method explained in the next segment of this section to give a sense score to each of the combination.
A query with n tokens and each token having k senses have (n^k) different combinations to check.

If a query has 3 tokens (A, B and C) and these tokens have 3, 2 and 4 senses respectively. Then there are total 3x2x4 = 24 possible combinations of sense to check for.

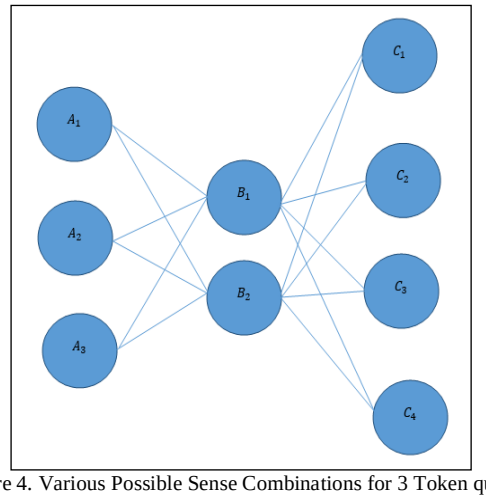

Figure 4. Various Possible Sense Combinations for 3 Token query

According to the Figure 5, $A_2B_2C_4$ is one such combination where 2nd sense of word A, 2nd sense of word B and 4th sense of word C is used.

$A_1B_1C_1$	$A_2B_1C_1$	$A_3B_1C_1$
$A_1B_1C_2$	$A_2B_1C_2$	$A_3B_1C_2$
$A_1B_1C_3$	$A_2B_1C_3$	$A_3B_1C_3$
$A_1B_1C_4$	$A_2B_1C_4$	$A_3B_1C_4$
$A_1B_2C_1$	$A_2B_2C_1$	$A_3B_2C_1$
$A_1B_2C_2$	$A_2B_2C_2$	$A_3B_1C_2$
$A_1B_2C_3$	$A_2B_2C_3$	$A_3B_1C_3$
$A_1B_2C_4$	$A_2B_2C_4$	$A_3B_1C_4$

Figure 5. List of all possible sense combinations

For the example query, possible combinations of senses are depicted in the Table 3.

TABLE III. POSSIBLE COMBINATIONS FOR EXAMPLE QUERY

मत (Opinion)	आधार (According)	फैसला (Decision)	Sense Score
A_1	B_1	C_1	S_1
A_1	B_1	C_2	S_2
:::	:::	:::	:::
A_2	B_1	C_1	S_{29}
:::	:::	:::	:::
A_5	B_{14}	C_2	S_{140}

Where A_i, B_j, C_k epresent the i^{th}, j^{th}, and k^{th} sense of the word मत (m t[2]), आधार (a□d□[2]a□r), फैसला (p□□□sla□) respectively and S_x is the sense score assigned to x^{th} combination.

2. *Sense Score Assignment*

In order to evaluate sense score for a given sense token combination, we follow the procedure depicted in the Figure-6. This algorithm considers all the pairwise matchings such that each element of a pair (a, b) belong to sense details of separate words. For every sense of each word gloss, hypernym, hyponym and synonym are considered to calculate a sense score for that pair.

For each pair (a, b) where a and b are both strings. We use the procedure elaborated below to assign scores.

For pair with a!=syn and b!=syn:
Common substring of length = len within a and b
*Score = Score + (len * (len + 1)) / 2*
Otherwise
Common substring of length = len within a and b
Score = Score + 2x l^{e}n3

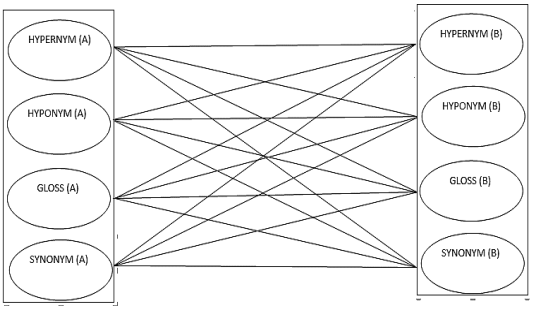

Figure 6. All possible pair of relations used to score a 2 token long query sentence.

The algorithm provides additional score in case the gloss of the one element of pair matches with the synonym directly, as it increases the probability of referred sense as the correct sense.

Pseudo Code 1: Sense Score Assignment

```
def scoring_method(string_a,string_b,factor):
score = 0
count = 0
   start_index = 0
   max_count = 0
   stri = ""
   string_a_list = string_a.split()
   string_b_list = string_b.split()
   iter1 = 0
   iter2 = 0
while iter1<len(string_a_list):
      iter2 = 0
      max_count = 0
      stri = ""
      common_words = ["||",",","कोई","किसी","वस्तु","हुआ","किया","जिसमें","जहाँ","किया","करते","-","जाता","जिसे","जिसके","करना","करने","जिसका","उसका","जाता","ऐसी","उसके","प्रकार","लाने","वाला","बहुत","हुए","होने","जाने","वाली","जिससे"]
      start_index = iter1
while iter2<len(string_b_list):
if iter1>=len(string_a_list):
break
```

```
if string_a_list[iter1] == string_b_list[iter2] and (string_a_list[iter1] not in common_words) and len(string_a_list[iter1])>0:
stri = stri + string_a_list[iter1] + " "
count = count+1
            iter1 = iter1+1
else:
if count>0:
                iter2 = iter2 - 1
                iter1 = start_index
count = 0
            stri = ""
if count>max_count:
            max_count = count
          iter2 = iter2 + 1
if factor == 1:
score = score + math.pow(max_count,3)
else:
score = score + math.pow(max_count,2)
count = 0
      iter1 = start_index + 1
if factor == 1:
score = score + math.pow(count,3)
else:
score = score + math.pow(count,2)
return score


Pseudo Code 2:
def lesk_algorithm(dataset):
   f = open("out.txt", "w")
   dataset_comb = []
   temp_list = []
for keys1 in dataset.keys():
      temp_list.append(keys1)
      dataset_comb.append(list(range(0,len(dataset[keys1]))))
print(dataset_comb)
print(len(list(itertools.product(*dataset_comb))))
   sense_comb = list(range(0,4*len(dataset.keys())))
   max_sense_score = 0
factor =0
for list_iter in list(itertools.product(*dataset_comb)):
      sense_score = 0
for first_sense_iter in sense_comb:
         first_word_index  = int(first_sense_iter/4)
         first_sense_index = first_sense_iter%4
         second_sense_iter = int(first_sense_iter/4)*4+4
while second_sense_iter<len(sense_comb):
if second_sense_iter >= len(sense_comb):
break
            second_word_index  = int(second_sense_iter/4)
            second_sense_index = second_sense_iter%4
if second_sense_index == 3 or first_sense_index == 3:
factor = 1
            sense_score = sense_score +
scoring_method(dataset[temp_list[first_word_index]][list_iter[first_word_index]][first_sense_index],dataset[temp_list[second_word_index]][list_iter[second_word_index]][second_sense_index],factor)
if sense_score > max_sense_score:
               max_sense_score = sense_score
               sense_result = list_iter

            f.write(str(list_iter)+" "+str(first_sense_iter)+" "+str(second_sense_iter)+" "+str(sense_score)+"\n")
            second_sense_iter = second_sense_iter + 1
print (sense_result)
print("max_score="+str(max_sense_score))
   i = 0
for it in sense_result:
print(temp_list[i]+"----->"+dataset[temp_list[i]][it][0])
      i=i+1
```

vi. Output (Word Sense Pair)

Dataset generated is parsed with the help of scoring criterion and each of the possible outcomes of the word sense combinations is evaluated. The one combination that yields best score (maximum score) is termed as the desired output of the query. Output for the algorithm is displayed in the form of word-sense pairs where each word is matched with their corresponding sense. Words and their senses with respect to combination associated with maximum sense score are displayed as output. For the example sentence output of the algorithm is depicted in Figure 7.

Output = Sense (max(s1,s2,s3…,s140))

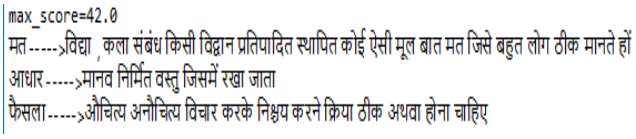

Figure 7. Output of the Example Sentence

3.2. Hindi WordNet Database

Typical dictionary has words associated with their meanings. Humans can dissect the sense of a sentence by simply looking into the meaning of its component words in the dictionary. Human brain is way more proficient in disambiguating a sentence while computer system cannot rely on such a dictionary. Thus computer system requires a way more advanced form of the dictionary in order to find relation between words and their different senses. One of such advanced structure is called WordNet which comprises of words and different kind of relations between them. That are synsets, gloss, synonym, hyponym, ontology sets etc. This system uses a Python based API to access Indian language WordNets that will help us accessing various wordnet related data like hypernym, hyponym, semantic relations between synsets, ontology nodes for 18 Indian languages, etc. "Pyiwn" python library is based on the हिन्दीशब्दतंत्र(□□[6]nd[2]i□[6]□bd[2]t[2]□̃t[2]r) (Hindi WordNet) designed by CFILT IIT Bombay. हिन्दीशब्दतंत्र(□□[6]nd[2]i□[6]

□bd[2]t[2]ɲ̃t[2]r) (Hindi WordNet) is a resource that binds various lexical and semantic relationships between the Hindi words.

हिन्दीशब्दतंत्र(□□[6]nd[2]i□[6]

□bd[2]t[2]ɲ̃t[2]r) (Hindi WordNet) is a dictionary with a large number of common Hindi words. It has the words stored in sets which are known as synsets. The synsets are a set of the words which can be identified by the same definition. For example,

The word मनुष्य(mnʊ[6]ʂj) (Manushya (Hindi), Human (English)) is present in a synset which has other words like:

मानव(ma□n ɦ[4]), आदमी(a:d[2]mi:[6]), इंसान(ɪ[6]ɔ̃sa□n), इन्सान(ɪ[6]nsa□n), इनसान(ɪ[6]nsa□n), मनुष्य(mnʊ[6]ʂj), मानुष(ma□nʊ[6]ʂ), मानुस(ma□nʊ[6]s), मनुज(mnʊ[6]ʤ), मनुष(mnʊ[6] ʂ), निदद्रु, मर्त्य(mrt[2j]), मर्दुम(mrd[2]ʊ[6] m)

All these words mean the same thing and can be identified by the definition:

"वहद्विपदप्राणीजोअपनेबुद्धिबलकेकारणसबप्राणियोंमेंश्रेष्ठहै औरजिसकेअंतर्गतहम, आपऔरसबलोगहैं"(ɦ[4]ɦ d[2]□[6]p d[2] pa□ni:[6] ʤo□ ə[5]pne□ □[4]□[6]d[2]d□[2]□[6]bl ke□ ka□rn sb pa□ni:[6]jo:ɔ̃ mo:ɔ̃ ʃe□r□□□[2] ɦɦ ɦɦ r ʤ □[6]s ke□ ə[5] ɔ̃t[2]rgt[2] ɦm, a:p ɦɦ r sb lo□g ɦɛɔ̃)

The word मनुष्य(mnʊ[6]ʂj) is a noun in Hindi, hence all the words in this synset are noun themselves. Similarly for any given word we can have different synsets for the different parts of speech that it can be used such as verbs, adjective or adverb.

For example the word खेल (Khel (Hindi), Play (English)) has 7 senses as a noun and 10 senses as verb and thus has different synsets for each of these senses in the WordNet, thus 17 in total.

We also have hypernyms and hyponyms for each word. Hyponymy shows the relationship between a generic term (hypernym) and a specific instance of it (hyponym). A hyponym is a word or phrase whose semantic field is more specific than its hypernym.

For the word खेल, its hypernym synset is:

खेलकूद(kʰe□lku:[6]d[2]), खेल-कूद(kʰe□l–ku:[6]d[2]), खेलकूद(kʰe□lku:[6]d[2]), मौज-मस्ती(m□□ʤ-mst[2]□[6]), मौजमस्ती(m□□ʤmst[2]□[6]), खेल(kʰe□l), क्रीड़ा(krɪ□[6]□a□), खिलवाड़(kʰ□[6]□[4]a□□), खेलवाड़(kʰe□□[4]a□□), आक्रीडन(a:krɪ□[6]□[2]n)

which is identified by the definition:

मनबहलानेयाव्यायामकेलिएउछल-कूद, दौड़-धूपयाऔरकोईमनोरंजककृत्य(mn b□la□ne□ ja□ □[4]ja□ja□m ke□ l□[6]j□□ ɦ[6]tɦɦ l- ku:[6]d[2] d[2]ɔ:ɽ-dʰ[2]u:[6]p ja□ ɦɦ r ko□i□[6] mno□r□ʤk krɪt[2]j)

The hyponyms of the word खेल would be: बिलियर्ड्स(b□[6]l□[6]jr□[2]s), अंताक्षरी(ə[5]̃t[2]a□□i□[6]), कबड्डी(kb□[2] ɦ[2] i□[6]) and so on.

We define gloss as the definition for each synset. From now on, gloss will be used to refer to any definition for a synset.

Some other notations:

1. Synset(word): Represents the set of all words present in the synset of the given word.

2. Synset(hypernym(word)): Represents the synset of the hyponym of a given word.

3. Gloss(synset(word)): Represents the gloss(definition) of the synset to which the word is associated.

In addition we're using indicnlp python library in order to perform stemming of the query sentence.i.e. Seperation of various forms of verbs into its root sentence for further processing

["बोलते"(bo□lt[2]e□),"बोलना"(bo□lna□),"बोले"(bo□le□),"बोलोगे"bo□lo□ge□] all four words mapped to a single verb बोल(bo□l) [to speak].

4. TIME COMPLEXITY OF PROPOSED SYSTEM

Let, query sentence is 'l' words long and it contains 'k_1' Stop words and 'k_2' word with no useful senses (words with no sense as noun, verb, adverb or adjective).

Tokens after the filtering and stemming step = l -k_1 -k_2 = l'

Assume, each of the l' tokens having $s_1, s_2, s_3 \ldots . s_l$, Senses associated with them.

Let,

Number of Total possible sense combination = s_1 x s_2 x…..x s_n = C

It is required to calculate sense score for n token long query.

Then total number of sense details pair =n x 4 x (n-1) x 4 operations= O (n^2)

Let, each sense detail pair is L lengths long. Time complexity require to compare a sense detail pair = O (L^2)
This complexity can be further improved by using any efficient string matching algorithm like KMP algorithm or Robin Karp algorithm etc.
Overall complexity of the algorithm = O (C x L^2 x n^2)

5. METHODOLOGY

Each of the sub steps explained in the architecture is elaborated here with an example sentence in Hindi.

Input Hindi Sentence: "आग में मेरा घर जल गया" (a:g me:ॅ me:ra: gʰr dʒl gja:)
आग(a:g) (Aag), में(me:ॅ) (me), मेरा, (me:ra:) (mera), घर(gʰr) (ghar), जल(dʒl) (jol), गया(gja:) (gayaa)
English Sentence: *"My house got burnt in the fire"*.
Figure 8 shows the Stop word in the given example sentence. Sentence consists of 'में'which is a stop word.

0	1	2	3	4	5
आग	में	मेरा	घर	जल	गया

Stop Word

Figure 8. Removing Stop Words from the Hindi Sentence

i) Filtered query after stop words elimination has been given in Figure 9.

0	1	2	3	4	5
आग	-	मेरा	घर	जल	गया

Figure 9. After Stop Word Elimination

ii) Stemming find the root word associated.
Example: For word 'जल'(dʒl) (Jal), root word is 'जलाना' (dʒla:na:) (Jalanaa).
Now, the Stemmed sentence has been given in Figure 10.

0	1	2	3	4	5
आग	-	मेरा	घर	जल,जलाना	गया

Figure 10. Stemmed Sentence

iii) POS tagging identifies the part of speech associate with each token in the filtered sentence. If there is no noun, adjective, verb, adverb sense associated with the token then algorithm simply reject that token as all the words belong to remaining part of speech is unisemous that does not need any disambiguation i.e. preposition, conjunction etc. Figure 11 and Figure 12 shows the rejection of unisemous word.

0	1	2	3	4	5
आग	-	मेरा	घर	जल,जलाना	गया
Noun		Pronoun	Noun	Verb,noun	Noun,verb

Unisemous word

Figure 11. Unisemous word

0	1	2	3	4	5
आग	-	-	घर	जलाना	गया

Figure 12. Stemmed Sentence

iv) Dataset generation step form a dataset that include all possible sense of the token left in the filtered sentence. A component of generated dataset is depicted in the Figure 13.

```
'जल': [
  [
    'नदी , जलाशय , वर्षा मिलने वाला द्रव पदार्थ पीने , नहाने , खेत सींचने काम आता ',
    '',
    'लवण मिला जल साबुन झाग बना पाता  || अलवणीय जल साबुन सरलता झाग बनाता
    'जल || पानी || नीर || अंबु || अम्बु || पय || वारि || आब || तोय || सलिल || पुष्कर ||
     अंभ || तामर || इरा || नीवर || यौनि || नार || कांड || काण्ड || ',
    'noun'
  ],
  [
    'पृथ्वी भाग जिसमें जल जल ढका ',
    'पृथ्वी कोई बड़ा भाग क्षेत्र  || ',
    'खारे पानी विशाल राशि चारों पृथ्वी स्थल भाग|
    'जलीय धरातल || जलीय-धरातल || जल || ',
    'noun'
  ],
  [
    'दूसरे लाभ हित देखकर मन कुढ़ना ',
    'खेद दुख करना  || ',
    '',
    'जलना || ईर्ष्या करना || द्वेष करना || डाह करना || कुढ़ना || दिल पर साँप लोटना || कलेजे पर साँप लोटना
    'verb'
  ],
  [
    'आग संपर्क अंगारे लपट रूप होना ',
    'रूप दूसरे रूप आना  || ',
    '',
    'जलना || सुलगना || दग्ध होना || दहना || सिलगना || अगियाना || अगिआना || ',
    'verb'
  ],
```

Figure 13. Dataset Format

v) Dataset is used to generate different sense combinations, each of which is evaluated using the designed algorithm.

Token	आग	घर	जल,जलाना	गया
No. of senses	3	10	11	3

Figure 14. Dataset Format

According to Figure 15, Total no. of sense combinations possible are = 3 x 10 x 11 x 3 = 990 combinations.

vi) Each of the 990 combinations are evaluated and one with the maximum score assigned as the output of the algorithm.

Python has been used to implement this algorithm. Python is an open source software tool and it is available in Web with its all modules and supporting libraries. The detailed software tools and libraries of this proposed system have been listed below.

i. Python 3.6: Basic environment for the application
ii. NLTK python library: Natural language processing library comprise of basic NLP methods
iii. CLTK python library: In order to inherit the stop word list for Hindi language
iv. Itertools python library: To successfully generate various sense combination in order to evaluate them for score.
v. Indicnlp python library: For the morphological processing of data to support stemming.

6. APPLICATIONS OF THE PROPOSED SYSTEM

The proposed algorithm is useful in systems, which includes analysis of a query to recommend related queries. In case user is not able to exactly describe the desired query, related results are suggested to user by this system. This system has many applications like in Pension Systems, where senior citizen can use words with which they are comfortable. Sentiment analysis of news headlines in order to predict the stock prices is another possible business application of the proposed algorithm. Positive and negative news have vast impact on the present trend for a stock. An automated system predicting the effect of a news on a stock price can use our proposed algorithm. The dissected sense can be used to various text summarization problems. Text summarization can use word sense disambiguation as one of their sub problems. Proposed algorithm can enhance the performance of text summarization algorithm and improve the efficiency of a search engine by understanding what the user is actually looking for after dissecting the input query.

7. FUTURE WORK

Proposed algorithm finds many combinations for sense tokens. It is not possible always to evaluate each of the combinations to get the correct sense as an output. Because even after numerous combinations that generate several senses, the correct or desired sense may not be derived as the output. Thus, we may incorporate some heuristic that can approximate our output to a near optimal result. Though, it is observed that such kind of queries does not occur too often in the natural language domain. Future work lies on identifying one or two heuristic methods based on genetic algorithm that may be used to provide a near optimal solution. Even deep learning based algorithm may be used in future to identify the correct sense of an ambiguous word in Hindi sentences. Hypernym, hyponym, gloss of each sense consist of various stop words and words with no available senses. Incorporating these words affect the performance of the algorithm. One possible solution to eliminate these words is to process them again by performing morphological analysis. But re-processing will increase the time complexity of the algorithm. As discussed earlier, this algorithm depends hugely on the semantic collection of words and their meanings provided by the WordNet. So the performance of the algorithm depends hugely on the WordNet. Dataset generation is a major step in the proposed algorithm. As it segregates only a portion of the data from WordNet, Due to which the database needs to be queried again and again. So an efficient execution of querying operation of WordNet can further enhance the performance of the proposed algorithm. The proposed system can be enhanced using machine learning algorithm or deep learning algorithms.

8. CONCLUSION

The proposed algorithm provides a word sense disambiguation solution for the Hindi language. The Proposed method rightly, gives pair of words and their evaluated senses as an outcome of the system. Novel scoring method used with Lesk alogorithm gives improved performance in disambiguating the ambiguous words for the Hindi language. Hindi WordNet database designed by CFILT IIT Bombay is very useful in solving Hindi word sense disambiguation problem. To improve the performance of Lesk algorithm, we have used the scoring method which assigns scores to the sense of the tokens depending on the combinations of sense tokens. Word gloss, hypernym, hyponym and synonym are very useful for calculating the sense score. The time complexity of the proposed algorithm is $O(n^2)$ where n is the number of tokens. Though algorithm imposes restriction on the length of the query sentence, it yields a good performance over constrained input data. Despite unavailability of large amount of ambiguous data involving polysemous words in Hindi language for testing, the proposed algorithm successfully dissects the sense for most of the test queries provided.

ACKNOWLEDGMENT

This research work is done at the Research Project Lab under Dept. of Computer Science and Engineering of National Institute of Technology (NIT), Durgapur. The authors would like to thank Dept. of Computer Science and Engineering, NIT, Durgapur, India for academically supporting this research work.

REFERENCES

[1] S. Malviya, R. Mishray, U. S. Tiwary, "Structural Analysis of Hindi Phonetics and A Method for Extraction of Phonetically Rich Sentences from a Very Large Hindi Text Corpus", 2016 Conference of The Oriental Chapter of International Committee for Coordination and Standardization of Speech Databases and Assessment Technique (O-COCOSDA), Bali, Indonesia, 2016, pp. 1-6.

[2] M.E. Lesk, "Automatic Sense Disambiguation Using Machine Readable Dictionaries: How to Tell a Pine Cone from an Ice Cream Cone", Proc. SIGDOC Conference, Toronto, Ontario, June, 1986, pp. 24-26.

[3] A.M. Butnaru and R. T. Ionescu, "ShotgunWSD 2.0: An Improved Algorithm for Global Word Sense Disambiguation," in IEEE Access, vol. 7, 2019, pp. 120961-120975.

[4] Preeti Dubey, "Word Sense Disambiguation in Natural Language Processing", JK Research Journal in Mathematics and Computer Sciences, vol. 1(1), 2018, pp.114-120.

[5] M. Y. Kang, T. H. Min and J. S. Lee, "Sense Space for Word Sense Disambiguation", 2018 IEEE International Conference on Big Data and Smart Computing (BigComp), Shanghai, 2018, pp. 669-672.

[6] R. Navigli, "Word sense disambiguation: A survey", ACM Comput. Surv., vol. 41, no. 2, Art. no. 10, 2009.

[7] A.G. Chifu and R. T. Ionescu, "Word sense disambiguation to improve precision for ambiguous queries", Central Eur. J. Comput. Sci., vol. 2, no. 4, 2012, pp. 398-411.

[8] M. N. Asim, M. Wasim, M. U. Ghani Khan, N. Mahmood and W. Mahmood, "The Use of Ontology in Retrieval: A Study on Textual, Multilingual, and Multimedia Retrieval", in IEEE Access, vol. 7, 2019, pp. 21662-21686.

[9] A.Sumanth and D. Inkpen, "How much does word sense disambiguation help in sentiment analysis of micropost data?", in Proc.WASSA, 2015, pp. 115-121.

[10] G. Xu, Z. Yu, H. Yao, F. Li, Y. Meng and X. Wu, "Chinese Text Sentiment Analysis Based on Extended Sentiment Dictionary", IEEE Access, vol. 7, 2019, pp. 43749-43762.

[11] M. Carpuat and D. Wu, "Improving statistical machine translation using word sense disambiguation", in Proc. EMNLP, 2007, pp. 61-72.

[12] X. Pu, N. Pappas, J. Henderson, and A. Popescu-Belis, "Integrating weakly supervised word sense disambiguation into neural machine translation", Trans. Assoc. Comput. Linguistics, vol. 6, 2018, pp. 635-649.

[13] J. N. Madhuri and R. Ganesh Kumar, "Extractive Text Summarization Using Sentence Ranking", International Conference on Data Science and Communication (IconDSC), Bangalore, India, 2019, pp. 1-3.

[14] L. Plaza, A. J. Jimeno-Yepes, A. Díaz, and A. R. Aronson, "Studying the correlation between different word sense disambiguation methods and summarization effectiveness in biomedical texts", BMC Bioinf., vol. 12, Art. no. 355, 2011.

[15] A.Taherkhani, A. Belatreche, Y. Li and L. P. Maguire, "A Supervised Learning Algorithm for Learning Precise Timing of Multiple Spikes in Multilayer Spiking Neural Networks", IEEE Transactions on Neural Networks and Learning Systems, vol. 29, no. 11, 2018, pp. 5394-5407.

[16] M. Usama et al., "Unsupervised Machine Learning for Networking: Techniques, Applications and Research Challenges", IEEE Access, vol. 7, 2019, pp. 65579-65615.

[17] L. Stanchev, "Semantic Document Clustering Using Information from WordNet and DBPedia", 2018 IEEE 12th International Conference on Semantic Computing (ICSC), Laguna Hills, CA, 2018, pp. 100-107.

[18] R. Pandit, S. Sengupta, S. K. Naskar and M. M. Sardar, "Improving Lesk by Incorporating Priority for Word Sense Disambiguation", Fifth International Conference on Emerging Applications of Information Technology (EAIT), Kolkata, 2018, pp. 1-4.

[19] S. Tang , F. Ma, X. Chen , P. Zhang, "Deep Chinese Word Sense Disambiguation Method Based on Sequence to Sequence", International Conference on Sensor Networks and Signal Processing (SNSP), 2018, pp. 498-503.

[20] A.Raganato, J. Camacho-Collados, R. Navigli, "Word Sense Disambiguation: A Unified Evaluation Framework and Empirical Comparison", 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, 2017, pp. 99-110.

[21] Q. P. Nguyen, A. D. Vo, J. C. Shin, P. Tran, C. Y. Ock, "Korean-Vietnamese Neural Machine Translation System With Korean Morphological Analysis and Word Sense Disambiguation", IEEE Access, vol. 7, 2019, pp. 32602-32616.

[22] S. Seifollahi, M. Shajari, "Word sense disambiguation application in sentiment analysis of news headlines: an applied approach to FOREX market prediction", Journal of Intelligent Information Systems, vol. 52(1), 2019, pp 57–83.

[23] Z. Vaishnav, P. S. Sajja "Knowledge-Based Approach for Word Sense Disambiguation Using Genetic Algorithm for Gujarati", ICTIS, vol. 1, 2018, pp. 485-494.

[24] S. N. Mohan Raj, S. S. Kumar, S. Rajendran, K. P. Soman, "Word Sense Disambiguation of Malayalam Nouns", Recent Advances in Computational Intelligence, Springer, 2019, pp 291-314.

[25] J. Singh, I. Singh, "Word Sense Disambiguation: Enhanced Lesk Approach in Punjabi Language", International Journal of Computer Applications, vol. 129(6), 2015, pp. 23-27.

[26] L. R. Pillai, V. Gangadharan, D. Gupta "A Combined Approach Using Semantic Role Labelling and Word Sense Disambiguation for Question Generation and Answer Extraction", Second International Conference on Advances in Electronics, Computers and Communications (ICAECC), Bangalore, 2018, pp. 1-6.

[27] H. Walia, A. Rana, V. Kansal, "A Naïve Bayes Approach for working on Gurmukhi Word Sense Disambiguation", 6th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO), 2017, Noida, pp. 432-435.

[28] M. Mahmood, M. Hourali "Semi-supervised approach for Persian word sense disambiguation", 7th International Conference on Computer and Knowledge Engineering (ICCKE), 2017, pp. 104-110.

[29] G. Sajini , Jagadish S. Kallimani, "Recognition and disambiguation of polysemy words in entered hindi documents", International Conference on Electrical, Electronics, Communication, Computer, and Optimization Techniques (ICEECCOT), 2017, pp. 1-8.

[30] S. Gopal , R. P. Haroon, "Malayalam word sense disambiguation using Naïve Bayes classifier", International Conference on Advances in Human Machine Interaction (HMI), Doddaballapur, 2016, pp. 1-4.

[31] X. Wang , X. Tang , W. Qu , M. Gu, "Word sense disambiguation by semantic inference", International Conference on Behavioral, Economic, Socio-cultural Computing (BESC), Krakow, 2017, pp. 1-6.

[32] S. Nazah, M. M. Hoque, Md. R. Hossain, "Word sense disambiguation of Bangla sentences using statistical approach", 3rd International Conference on Electrical Information and Communication Technology (EICT), 2017, pp. 1-6.

[33] S. Banerjee, T. Pedersen, "An adapted lesk algorithm for word sense disambiguation using wordnet", Computational Linguistics and Intelligent Text Processing, Springer, 2002, pp 136-145.

AUTHORS

Praffullit Tripathi is a Software Engineer in Qualcomm Company. He has completed his B.Tech-IT degree from National Institute of Technology, Durgapur, India. He has sucessfully completed his summer internship at Qualcomm Company. He has a good interest in Implementation of Natural Language Processing based Applications like Word Sense Disambiguation, Query-Response Model, Question Answering System, Sentiment Analysis, etc.

Prasenjit Mukherjee has 12 years of experience in academics and industry. He was a fulltime Ph.D. Research Scholar in Computer Science and Engineering in the area of Natural Language Processing from National Institute of Technology (NIT), Durgapur, India under the Visvesvaraya PhD Scheme from 2015 to 2019. Presently, He is working as a Data Scientist under Analytics and IT Department,RamanByte, Pune Institute of Business Management, Pune, Maharashtra, India.

Manik Hendre has 5 years of experience in academics and industry. He is a Ph.D. research scholar at the University of Pune. He is currently working as a Data Scientist in RamanByte Pune. His research areas include Biometrics Image Processing, Machine learning and Data Analytics.

Dr. Manish Godse has 25 years of experience in academics and industry. He holds Ph.D. from Indian Institute of Technology, Bombay (IITB). He is currently working as an Industry Professor and IT Director in the PIBM, Pune in the area of Artificial Intelligence and Analytics. His research areas of interest include automation, machine learning, natural language processing and business analytics. He has multiple research papers indexed at IEEE, ELSEVIER, INDERSCIENCE, etc.

Dr. Baisakhi Chakraborty received the PhD. degree in 2011 from National Institute of Technology, Durgapur, India in Computer Science and Engineering. Her research interest includes knowledge systems, knowledge engineering and management, database systems, data mining, natural language processing and software engineering. She has several research scholars under her guidance. She has more than 60 international publications. She has a decade of industrial and 14 years of academic experience.