An adaptive approach for word sense disambiguation for Hindi language

An adaptive approach for word sense disambiguation for Hindi language

Vimal Kumar Soni†, Dinesh Gopalani‡ and M C Govil§
Department of Computer Science and Engineering
Malaviya National Institute of Technology, Jaipur, India
E-mail: †ruvimals@gmail.com, ‡dgopalani.cse@mnit.ac.in, §govilmc@gmail.com

Abstract. Word sense disambiguation has been a classical problem and it is still considered as a big challenge for NLP researchers. Word vectors have shown remarkable improvements from baseline performances for various NLP challenges. An adaptive approach is developed to resolve the ambiguity at word level using word vectors and presented in this paper. CBOW and Skip-gram models of Word2Vec are used to generate the word embeddings. An input sentence is split into two sets: The first set contains all possible ambiguous words and the second set is constituted from all the unambiguous words. The words from the first set are processed through IndoWordNet to collect all the senses of respective words, and such senses are stored in sense-dictionary. An adaptive approach is applied to find out the exact interpretation of the ambiguous word from the sense dictionary for each word from the set of ambiguous words with the context words from the set of unambiguous words. The proposed approach is evaluated on a large scale Hindi corpus and claimed better results than previous attempts.

1. Introduction

Natural Languages are ambiguous by default. A sentence can be ambiguous due to one or many ambiguous words or due to the order of the words. Sometimes the punctuation also plays a vital role in multiple meaning of a sentence. Word Sense Disambiguation (WSD) is used to find out the correct and relevant meaning (sense) of a word which can have multiple meaning in the sentence. Without applying Word Sense Disambiguation, a sentence can be interpreted in more than one meaning. Out of these multiple meanings of the sentence, at least one of the meanings will be always correct however sometimes more than one meaning are also sensible. The WSD is counted in initial steps of language processing, as without it, sense of a sentence can be misinterpreted [1].

In normal human communication, humans perceive secondary information like context from the environment and the sender of the information. With the support of such information, humans can perceive the right meaning of a word out of multiple meanings easily. However such crucial information is not available in the text while processing it by machines which results to develop the correct meaning of the sentence is a challenging task.

WSD is defined as the “ability to identify the meaning of words in context in a computational manner” by Navigli[3]. A word may act ambiguously at any place in a sentence or the word may act ambiguously in a specific role or position in the sentence. In the first case, it is required to perform WSD on all words in a sentence however in the later one WSD is applied only on the specific word. Based on this, there are two classifications- WSD for all words in a sentence and WSD for Lexical-sample.

Researchers are taking advantage of huge web crawled data in the form of corpus and applying different deep/shallow learning methods to generate word embeddings or word vectors. A word embedding or

word vector is a learned representation for words where the words with similar semantics have related and close representation.

In humans, the meaning of a word is associated with certain neurons and weights among their interconnection in the brain. Similar words with related meaning form logical groups associated together. The word vectors also follow the same principle. The dimensions of such vectors vary from 50-600 as per the requirement of the application and availability of computing resources. These word vectors are represented using real numbers to imitate the behaviour of a word and have performed extremely well in various NLP tasks. There are various approaches to generate word embeddings like Hard and Soft Clustering models, Mathematics based counting models and Neural-Network based models.

The word vectors generated using neural-network based models specifically word2vec and fastText outperformed vectors generated by other models in most of the NLP tasks. Most of the research works have been carried out in the English language for the generation of word vectors for the English Language. Researchers have focused more on certain languages like Arabic, Chinese, French Japanese and Russian. However, few attempts like Polyglot have been made to develop word vectors in a group for multiple languages in one go. For the Hindi Language too, continuous efforts are in progress to develop quality word vectors. This paper is an attempt to create word embeddings using word2vec and use these word embeddings for word sense disambiguation.

The next section presents an extensive survey of the work done in the domains of WSD and Word Representation respectively. Section III describes the proposed work and after that, the methodology is explained in Section IV. Experimental results are discussed in Section V. The last section covers the conclusion and future work in a nutshell.

2. Related Work

2.1. Word Sense Disambiguation:

The automatic WSD is considered as a classical problem in NLP domain from the starting of automated language processing. With the genesis of automatic machine translation, WSD was assumed as a sub-task. Numerous research attempts have been made for the English WSD, however, for the Hindi language, WSD challenge is still a hot topic.

Ide et al. discussed WSD in detail and termed it as “AI-Complete”. That is, “a problem which can be solved only by first resolving all the difficult problems in artificial intelligence (AI), such as the representation of common sense and encyclopedic knowledge” [1]. Agirre and Stevenson [2] provided a thorough analysis of finding clues through the syntactic and semantic analysis of a sentence in 2007. Navigli relied on knowledge-based approaches and also mentioned that supervised methods outperform unsupervised methods [3]. Zhong et al. took designed a system to take leverage of POS tagging, context window of 5 surrounding words and local collocations to solve the challenge of WSD [4].

Taghipour et al. proposed a framework using word embeddings and previous model with changing the context window to 7 surrounding words and 11 local collocations around the word [5]. Rothe and Hinrich used synsets and word embeddings to propose a model with an assumption that a word was a sum of its lexeme [6]. Chen et al. presented a unified model which developed one representation per sense and outperformed other systems [7]. They also developed domain-specific embeddings focusing on Sports and Finance domain.

Research Work in for WSD in the Hindi language has not been as vast as it is in the English Language. The first attempt is considered by Sinha et al. who applied a statistical technique to disambiguate words with the help of wordnet [8] in 2004. Mishra et al. proposed a solution using a classifier which learned from unlabeled Hindi corpus [9] in 2009. Singh et al. proposed an algorithm using hypernym, hyponym, holonym and meronym relations to solve the ambiguity and proved that hyponyms could help.[10]. Agarwal and Bajpai et al. explored the disambiguation of nouns using wordnet in 2014 [11].

Patel and Bhatt proposed a solution based on Hierarchical clustering technique [12]. In 2015, Tayal et al. presented a solution using Fuzzy C-Means Clustering to create sense topics with which they resolved the ambiguity of the word [13]. In 2016, Singh et al. take a step forward in the direction to resolve ambiguity by developing annotated sense corpus [14]. In 2018, Athaiya et al. presented a genetic new

approach using genetic algorithm with the feature of a dynamic window subject to the condition of availability of topic in the neighbourhood of the ambiguous word [15]. Sharma and Joshi developed a system which used the knowledge-based LESK algorithm and Wordnet [16]. Kumari and Lobiyal proposed a solution to use word embeddings to resolve the ambiguity [17].

2.2. Distributed Representation of Words

The one-hot representation is one where each word is represented as a sparse vector of the same size of the vocabulary and only one value is set 1. The method of representing a word in one-hot vector representation suffers from the curse of dimensionality as the one-hot vectors are quite large yet sparse. Distributed representation of words (Word Embeddings) resolved this challenge by representing the vectors in limited dimensions with real values. Consider a vocabulary created using the following two documents: D1 and D2:

D1: यह आम रास्ता नही है|

D2: मुझे आम खाना पसन्द है|

So after removing stop-words, the resultant Vocabulary V is.

V = { 'आम', 'रास्ता', "खाना" , "पसन्द" }

Examples of one-hot representation and distributed representation are mentioned in Table-1 and Table-2 respectively.

Table 1. One-hot representation of words

Word	Word Vector			
आम	1	0	0	0
रास्ता	0	1	0	0
खाना	0	0	1	0
पसन्द	0	0	0	1

Table 2. Distributed representation of words

Word	Word Vector
आम	{0.7766, -0.3528, 0.1335, -0.3631, -0.5277}
रास्ता	{0.1507, 0.0229, -0.2650, -0.3493, 0.4425}
खाना	{0.647, 0.2823, -0.3315, 0.6313, -0.7752}
पसन्द	{0.3071, -0.2923, 0.5023, 0.9334, -0.5414}

In 1986, Rumelhart et al. introduced the concept of backpropagation [18]. Bengio et al. introduced word embeddings in 2003 with their neural probabilistic language model [19]. This model was developed using a feed-forward neural network to forecast the next word. Collobert and Weston fetch semantic information from a large corpus in the form of word embeddings in their unified architecture [20]. Mikolov et al. in 2013 brought a kind of revolution in the research world by proposing CBOW and Skip-gram Models and word2vec tool [21].

These two models did not only generate effective word vectors on the basis of the semantics of the word but also were able to manipulate the meaning of the word if its vector is changed after performing vector operations on it. The CBOW takes surrounding words and forecasts the target word, on the contrary, the Skip-gram model accepts a word and predicts its neighbour/context words. Joulin et al. used sub-word information like morphological variations to generate improved vectors [23]. Different Embedding Models and their objective functions are presented and diccussed in detail in [24].

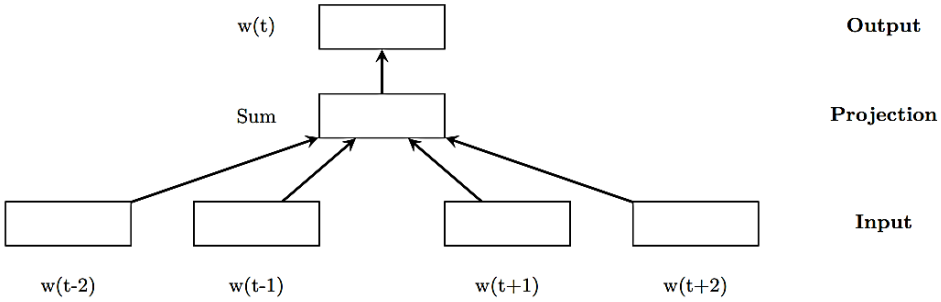

Figure 1. Continuous bag-of-words (CBOW) Model [21]

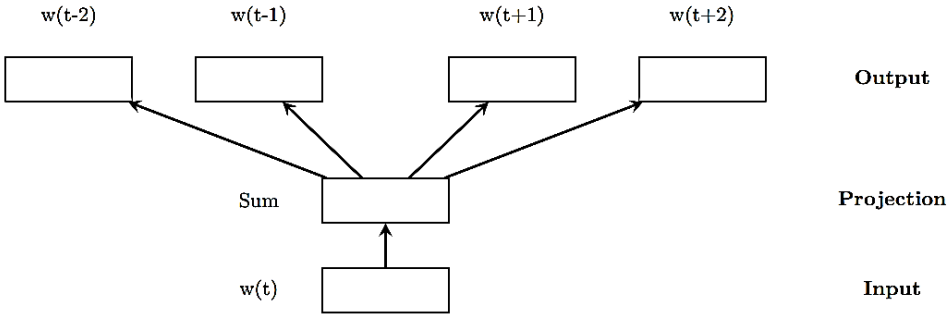

Figure 2. Skip-Gram Model [21]

In this work, we have used variations in tuning parameters of word2vec. Three tuning parameters are considered for such variations are– 1. Vector Dimensions, 2. Window Size and 3. Negative Sampling.

3. Proposed Approach

First, the corpus is collected using various methods and then it was pre-processed using standard methods to clean and make it ready to use. This pre-processing includes Tokenization and Normalization with the following steps:

- Remove numbers and special characters
- Replacing punctuation with white spaces
- Stripping white spaces
- Filtering stop-words

Siddiqi et al. have presented an extensive list of more than 800 stop-words in their research work [25]. This list has been used in this research work to filter stop-words from the corpus as well as from the test sentence. Afterwards, the word vectors are generated using this pre-process corpus. These word vectors are later used in the second algorithm to map words from the given test sentence to their vector forms. The first algorithm WSD_VectorGeneration is shown below:

Algorithm : WSD_VectorGeneration (Corpus, D, W, NS)

Input: Corpus, Tuning Parameter
Output: Word Vector Repository

1. Preparation of Corpus
2. Pre-processing of corpus
3. Set tuning parameters
4. Generate Word Vectors and store in Vector Repository

Algorithm 1: WSD_VectorGeneration

The Second Algorithm WSD_VectorProcessing written inline with Python language is shown below.

Algorithm : WSD_VectorProcessing (Vector_Dictionary)

Input: Vector_Dictionary, Test_Sentence
Output: The Best One Sense out of multiple meanings for the ambiguous word which suits the best semantically in the Test Sentence

```
1.  Input Test Sentence
2.  Pre-processing and cleaning of Test Sentence
3.  Tokenization to create word list WL
4.  for each word in WL do:
        if ambiguous(word)==True:
            AS.append(word)
        else
            US.append(word)
        end if
    end for
5.  Create Empty Context Dictionary CD
6.  for each word in AS do:
        context = US + AS - word
        CD[word]= [context]
    end for
7.  Create Context Vectors Dictionary CVD
8.  for each word in CD do:
        temp_vec=0
        for each contextword in CD[word] do:
            temp_vec=temp_vec+word2vec(contextword)
        CVD[word]= temp_vec / len(CD[word])
        end for
    end for
9.  Create Empty Sense Dictionary SD
10. for each word in AS do:
        senses = IndoWordNet_FetchSense(word)
        SD[word]= [senses]
    end for
11. Create Sense Vectors Dictionary SVD
12. for each word in SVD do:
        sense_vec=[]
        for each sense in SD[word] do:
            temp_vec=0
            for each part in sense do:
                sense_vec.append=word2vec(part)
            end for
        SVD[word] = sense_vec
        end for
    end for
13. Create Empty Reference Vector Dictionary RVD
14. for each word in SVD do:
        temp=id=0
            for each vector in SVD[word] do:
                temp= CVD[word]-(sum(SVD[word])-vector)
                RVD[[word,id]]= vector + temp
            end for
    end for
15. Create Test Sentence Vector TSV
16. Generate Sentence Decode Vectors' Matrix SDV
17. Choose vector from SDV such that it has minimum
    cosine distance from TSV
18. The id represents the index of the word with
    the best applicable sense w.r.t. Input Sentence
```

Algorithm 2: WSD_ VectorProcessing

A given test sentence is segregated to two sets: The first set contains all possible ambiguous words from the test sentence and the second set is constituted from all the unambiguous words. The words from the first set are processed through IndoWordNet to collect all the senses of respective words, and such senses are stored in sense-dictionary. An adaptive approach is applied to discover the exact sense of the ambiguous word from the sense dictionary for each word from the set of ambiguous words with the help of context words from the set of unambiguous words. Finally, the Cosine Similarity is used as a standard measure to evaluate the semantic relation between the word vectors.

4. Methodology

This system works in two phases. In the first phase, the corpus is pre-processed and Word Vectors are generated from this pre-processed corpus. The corpus used to implement this paper is formed using two different parts: first, Indian Corpora from NLTK and second, Web crawled data from Wikipedia. After forming the corpus by amalgam, it was pre-processed using the steps mentioned in the previous section. To generate word vectors from this corpus, we have to finalize the tuning parameters of CBOW and Skip-gram models.

The variations in tuning parameters are arranged in three different combinations for both CBOW and Skip-gram. In the first combination, the dimensions of vectors are fixed as 50, with a window size of 15 words and no negative sampling. The second combination consists of the vector dimensions as 150, window size as 10 words and with a negative sampling of 5 words. The vector dimensions and negative sampling size are increased to 200 and 10 respectively while the window size is decreased to 5 in the last combination. 100 and 150. With this setup, the word vectors are generated. The same process is shown in Figure-3.

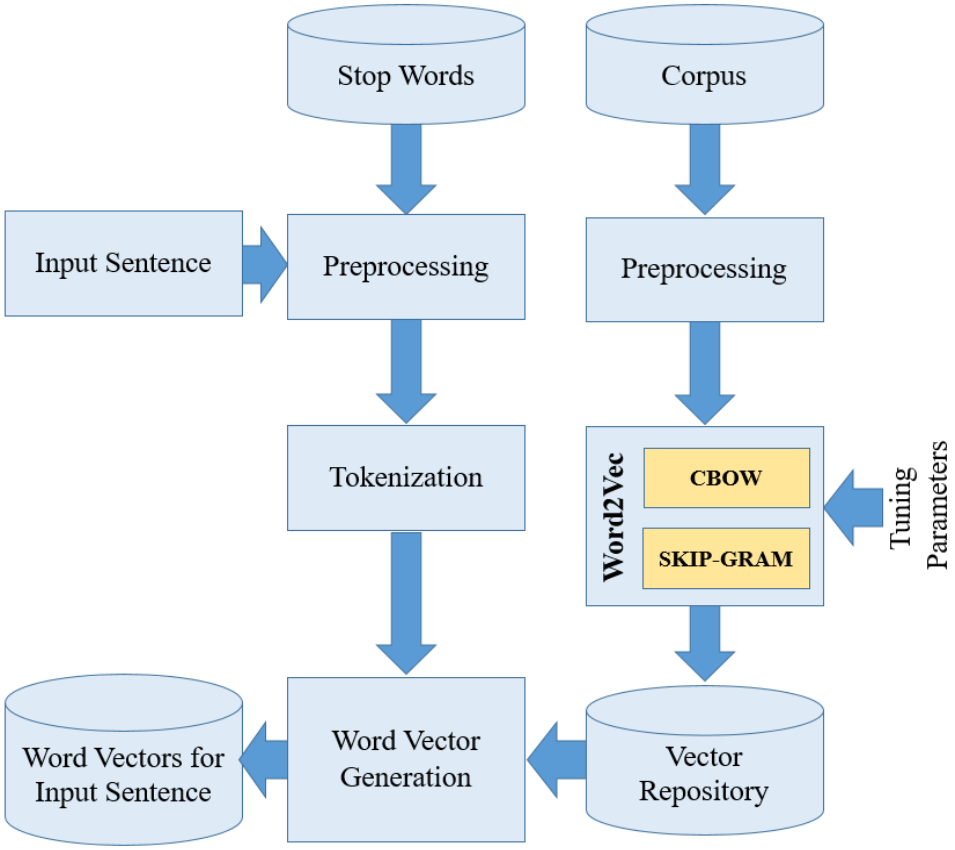

Figure 3. Generation of word vectors

The steps of the first algorithm are self-explanatory, however, the second algorithm is explained below due to its complexity:

1. Input Test Sentence

2. Preprocessing and cleaning of Test Sentence

3. Tokenization

4. Segregation of Words to Ambiguous and non-ambiguous word sets

$$AS = \{w_1, w_2, \dots, w_A\}$$

Where AS represents the Ambiguous Word Set, and
A denotes the total number of ambiguous words in the set

$$US = \{w_1, w_2, \dots, w_U\}$$

Where US represents the Unambiguous Word Set, and
U denotes the total number of ambiguous words in the set

5. Construction of Context Dictionary from the Set of unambiguous words with respect to each ambiguous word from Ambiguous Word Set

$$CD = \{w_1 : [w_{11}, w_{12}, \dots, w_{1p}], w_2 : [w_{21}, w_{22}, \dots, w_{2q}], \quad \dots\ w_A : [w_{A1}, w_{A2}, \dots, w_{Ar}]\ \}$$

Where CD represents the Context Dictionary.
p represents the total number of context words for word-1
q denotes the total number of context words for word-2
r symbolizes the total number of context words for word-A

6. Construction of Context Vectors Dictionary from Context Dictionary with a 1 to 1 mapping of Ambiguous Word to its Context Vector.

$$CVD = \{\ w_1 : CV_1, w_2 : CV_2, \dots \quad w_A : CV_A, \}$$

Where CVD represents the Context Dictionary.
CV1 represents the context vectors for word-1 and
CVA represents the context vectors for word-A
And A represents the total number of words in the dictionary.

7. Fetch sense of each word of the ambiguous set using Indowordnet. The Sense Dictionary is then updated by entering the word and its fetched (multiple) meanings from the Indowordnet.

Since Ambiguous word set has a total of "A" words so the Sense dictionary will also have total "A" entries. However, there will be multiple entries of senses against each word.

$$SD = \{w_1 : [S_{11}, S_{12}, \dots, S_{1X}], w_2 : [S_{21}, S_{22}, \dots, S_{2Y}], \dots w_A : [S_{A1}, S_{A2}, \dots, S_{AZ}]\}$$

Where SD represents the Sense Dictionary,
A is the total number of words in the SD
S_{1X} represents the X^{th} Sense for word-1 and
S_{AZ} represents the Z^{th} Sense for word-A.

8. Construction of Sense Vectors Dictionary for each sense/ meaning for each Ambiguous Word with the help of Sense Dictionary

$$SVD = \{w_1 : [SV_{11}, SV_{12}, \dots, SV_{1X}], w_2 : [SV_{21}, SV_{22}, \dots, SV_{2Y}] \\ \dots w_A : [SV_{A1}, SV_{A2}, \dots, SV_{AZ}]\ \}$$

Where SVD represents the Sense-Vector Dictionary.
SV_{1X} represents the sense vector of word-1 for X^{th} sense
SV_{AZ} represents the sense vector of word-A for Z^{th} sense

9. Calculation of Reference Vectors with the help of Context Vector and Sense Vector using following equation-

$$R_{ij} = SV_{ij} + \sum_{k=1,k\neq j}^{n_i} CV_i - SV_{ik}$$

Where R_{ij} denotes reference vector of the i^{th} word for j^{th} Sense,
SV_{ij} represents the Sense Vector of i^{th} word for j^{th} Sense
CV_i refers to the Context Vector of i^{th} word, and
n_i symbolizes the total number of senses for i^{th} word

10. Generation of Test Sentence Vector with all words

$$TSV = V_{w_1} + V_{w_2} + \cdots + V_{w_m}$$

Where TSV is a composite vector of The Test Sentence, and
V_{w_1} represents the Vector of word1 and
V_{w_m} denotes the vector of word m.
where m is the length of Sentence in terms of words.

11. Generation of Sentence Decode Vectors- It is the same as generating TSV except that to create this, we put the Reference Vector instead of Vectors of ambiguous words. Since ambiguous words will have more than one meaning so this will result in a matrix of multiple vectors. This SDV Matrix, each column represents one word and one row represents a complete sentence with one meaning of the ambiguous word and thus this SDV matrix contains all the possible interpretations of the sentence which might be meaningful or meaningless.

$$SDV_{ij} = V_{w_1} + R_{ij} + \cdots + V_{w_m}$$

Where m is the number of words in Test Sentence after tokenization.

12. Selection of the reference vector which is semantically most close to Vector of target Sentence can be done using

$$Sim(TSV, SDV_{ij}) = argmax\ sim(TSV, SDV_{ij})$$

$$Sim(TSV, SDV_{ij}) = argmax\ \frac{TSV.\ SDV_{ij}}{\|TSV\|.\|SDV_{ij}\|}$$

13. The sense related to the vector (SDV_{ij}) with a minimum cosine distance from TSV is the best interpretation of the ambiguous word.

To explain the process further, we have taken a test sentence with an ambiguous word – "आम".

Test Sentence 1:

बड़े शहरों में तो यह समस्या आम इंसान की है |

After removing stop words and cleaning:

बड़े शहरों समस्या आम इंसान

Context Vectors:
For each ambiguous word, we need to create one context vector.
For Test Sentence 1:

CV_1 = (V(बड़े) + V(शहरों)+ V(समस्या) + v(इंसान))

The Glosses of the target word “आम”, fetched from IndoWordNet are shown in Table-3.

Table 3. Glosses of word - "आम" from IndoWordNet

Gloss No.	Synset ID	Gloss	POS
Gloss1	3462	एक फल जो खाया या चूसा जाता है	noun
Gloss2	3468	जिसमें कोई विशेषता न हो या अच्छे से कुछ हल्के दरज़े का	adjective
Gloss3	3463	गर्म देशों में पाया जाने वाला एक बड़ा, सदाबहार पेड़ जिसके रसीले फल खाए या चूसे जाते हैं	noun
Gloss4	3469	प्रायः सभी व्यक्तियों, अवसरों, अवस्थाओं आदि में पाया जाने वाला या उनसे संबंध रखने वाला	adjective
Gloss5	2283	जो पका हुआ न हो	adjective
Gloss6	2697	जो कम पका हुआ हो	adjective
Gloss7	6253	एक रोग जिसमें शरीर से चिकना, सफ़ेद, लसदार मल बार-बार निकलता है	noun
Gloss8	8466	एक प्रकार का लसदार चिकना सफेद मल जो अन्न न पचने से उत्पन्न होता है	noun
Gloss9[a]	Blank	Blank	adjective

[a.] 9 Synset fetched from IndoWordNet for "आम" including 1 blank Synset.

The process of WSD_ VectorProcessing() up to the creation of Reference Vector is depicted in Figure4.

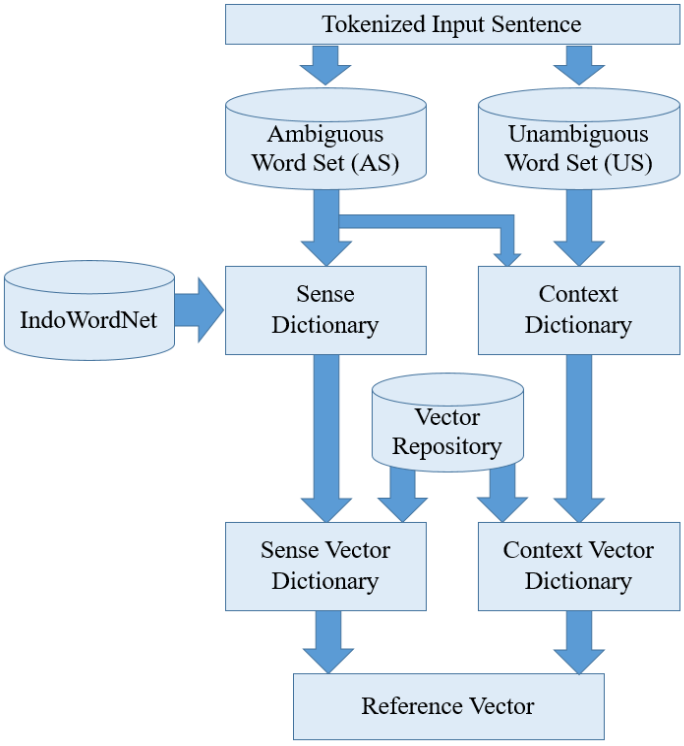

Figure 4. Generation of Reference Vector by Vector Processing

Sense Vectors:

$SV_{आम1}$ = (V(फल) + V(खाया)+ V(चूसा))/3

$SV_{आम2}$ = (V(विशेषता) + V(अच्छे) + V(हल्के) + V(दरज़े))4

$SV_{आम3}$ = (V(गर्म) + V(देशों) + V(पाया) + V(बड़ा) + V(सदाबहार) + V(पेड़) + V(रसीले) + V(फल) + V(खाए)+V(चूसे))/10

$SV_{आम4}$ = (V(प्राय) + V(व्यक्तियों) + V(अवसरों) + V(अवस्थाओं) + V(पाया) + V(संबंध))/6

$SV_{आम5}$ = (V(पका))/1

$SV_{आम6}$ = (V(कम) + V(पका))/2

$SV_{आम7}$ = (V(रोग) + V(शरीर) + V(चिकना) + V(सफ़ेद) + V(लसदार) + V(मल) + V(बार) + V(चिकना) + V(निकलता))/9

$SV_{आम8}$ = (V(लसदार) + V(चिकना) + V(सफ़ेद) + V(मल) + V(अन्न) + V(पचने) + V(उत्पन्न))/7

$SV_{आम9}$ = Blank

Reference Vectors:
Reference Vectors are to be created for each sense for an ambiguous word. There are 9 Synsets mentioned for the ambiguous word – "आम", however only 8 glosses/senses are available, therefore only 8 reference vectors are generated.

$$R_{आम1} = SV_{आम1} + \sum_{k=1,k\neq1}^{8} CV_1 - SV_{आमk}$$

$$R_{आम2} = SV_{आम2} + \sum_{k=1,k\neq2}^{8} CV_1 - SV_{आमk}$$

$$R_{आम3} = SV_{आम3} + \sum_{k=1,k\neq3}^{8} CV_1 - SV_{आमk}$$

$$R_{आम4} = SV_{आम4} + \sum_{k=1,k\neq4}^{8} CV_1 - SV_{आमk}$$

$$R_{आम5} = SV_{आम5} + \sum_{k=1,k\neq5}^{8} CV_1 - SV_{आमk}$$

$$R_{आम6} = SV_{आम6} + \sum_{k=1,k\neq6}^{8} CV_1 - SV_{आमk}$$

$$R_{आम7} = SV_{आम7} + \sum_{k=1,k\neq7}^{8} CV_1 - SV_{आमk}$$

$$R_{आम8} = SV_{आम8} + \sum_{k=1,k\neq8}^{8} CV_1 - SV_{आमk}$$

Now after obtaining these Reference Vectors for the ambiguous word, the Sentence Decode Vectors are computed as mentioned below.

$$SDV_{\text{आम}1} = V_{\text{बड़े}} + V_{\text{शहरों}} + V_{\text{समस्या}} + R_{\text{आम}1} + V_{\text{इंसान}}$$
$$SDV_{\text{आम}2} = V_{\text{बड़े}} + V_{\text{शहरों}} + V_{\text{समस्या}} + R_{\text{आम}2} + V_{\text{इंसान}}$$
$$SDV_{\text{आम}3} = V_{\text{बड़े}} + V_{\text{शहरों}} + V_{\text{समस्या}} + R_{\text{आम}3} + V_{\text{इंसान}}$$
$$SDV_{\text{आम}4} = V_{\text{बड़े}} + V_{\text{शहरों}} + V_{\text{समस्या}} + R_{\text{आम}4} + V_{\text{इंसान}}$$
$$SDV_{\text{आम}5} = V_{\text{बड़े}} + V_{\text{शहरों}} + V_{\text{समस्या}} + R_{\text{आम}5} + V_{\text{इंसान}}$$
$$SDV_{\text{आम}6} = V_{\text{बड़े}} + V_{\text{शहरों}} + V_{\text{समस्या}} + R_{\text{आम}6} + V_{\text{इंसान}}$$
$$SDV_{\text{आम}7} = V_{\text{बड़े}} + V_{\text{शहरों}} + V_{\text{समस्या}} + R_{\text{आम}7} + V_{\text{इंसान}}$$
$$SDV_{\text{आम}8} = V_{\text{बड़े}} + V_{\text{शहरों}} + V_{\text{समस्या}} + R_{\text{आम}8} + V_{\text{इंसान}}$$

Out of these 8 Sentence Decode Vectors, the closest vector to the Test Sentence Vector should be used for the translation of the ambiguous word.

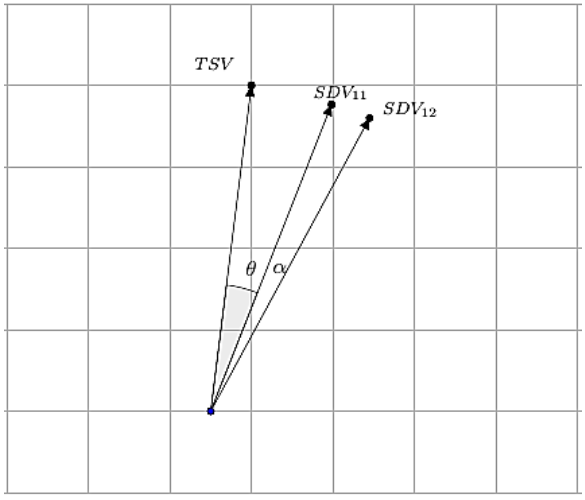

Figure 5. Cosine Distance Between Test Sentence Vector and Sentence Decode Vectors

To check the similarity, the cosine distance is calculated between Test Sentence Vector (TSV) and each of Sentence Decode Vector (SDV). To calculate similarity, we need to compute the TSV vector, which can be done like this:

$$TSV = V_{\text{बड़े}} + V_{\text{शहरों}} + V_{\text{समस्या}} + V_{\text{आम}} + V_{\text{इंसान}}$$

All the vectors for TSV are taken directly from the vector repository. Now the similarity between TSV and SDV can be calculated by this equation:

$$sim(TSV, SDV_i) = \frac{TSV.SDV_i}{\|TSV\|\|SDV_i\|}$$

Where $\|TSV\|$ denotes Euclidean norm of TSV and $\|SDV_i\|$ refers to Euclidean norm of SDV_i

If two vectors are identical than the cosine value will be zero and orthogonal relation between vectors indicates there is no match. The value of angle reaching to zero (or cosine value reaching to one) specifies that the meanings of two words are quite close to each other. The similarity scores are mentioned in the following Table-4.

Table 4. Cosine Similarity Score

Gloss/ Sense Id	Senses	Similarity Equation	Similarity Score	Rank
1	एक फल जो खाया या चूसा जाता है	$sim(TSV, SDV_1)$ =	88.72	4
2	जिसमें कोई विशेषता न हो या अच्छे से कुछ हल्के दरज़े का	$sim(TSV, SDV_2)$ =	91.29	2
3	गर्म देशों में पाया जाने वाला एक बड़ा, सदाबहार पेड़ जिसके रसीले फल खाए या चूसे जाते हैं	$sim(TSV, SDV_3)$ =	89.84	3
4	प्रायः सभी व्यक्तियों, अवसरों, अवस्थाओं आदि में पाया जाने वाला या उनसे संबंध रखने वाला	$sim(TSV, SDV_4)$ =	93.48	1
5	जो पका हुआ न हो	$sim(TSV, SDV_5)$ =	79.35	5
6	जो कम पका हुआ हो	$sim(TSV, SDV_6)$ =	78.62	6
7	एक रोग जिसमें शरीर से चिकना, सफ़ेद, लसदार मल बार-बार निकलता है	$sim(TSV, SDV_7)$ =	74.91	8
8	एक प्रकार का लसदार चिकना सफेद मल जो अन्न न पचने से उत्पन्न होता है	$sim(TSV, SDV_8)$ =	75.54	7
9[a]	Blank	-	-	-

[a] 9 Synset fetched from IndoWordNet for "आम" including 1 blank Synset.

Further, we tried our approach with other meanings of the same word – "आम" with two more Test Sentences:

Test Sentence 2:

मुझे फलों में आम बड़े पसंद हैं |

Test Sentence 3:

पाचनतंत्र दुरुस्त न होने से भोजन अधपचा रहने की समस्या को आम कहते हैं।

The Test Sentence-2 has two words common with Test Sentence-1 i.e. "बड़े" and "आम". Similarly, the Test Sentence-3 has "समस्या" and "आम" common with Test Sentence-1. Now, we need to create Context Vectors for Test Sentence 2 and Test Sentence 3:

CV_2 = (V(फलों) + V(बड़े) + V(पसंद))

CV3 = (V(पाचनतंत्र) + V(दुरुस्त) + V(भोजन) + V(अधपचा) + V(समस्या))

The sense vectors are the same because of the same ambiguous word. However, the Reference Vectors as well as Sentence Decode Vectors will be different for each sentence.

5. Results

The closest senses of the ambiguous word-"आम" computed by our approach for each of the three above-mentioned Test Sentences are shown in Table-5.

Table 5. Sense disambiguation of the word-"आम"

Sentence	Total Ambiguous Senses	Number of Close Senses	Most Close Sense
बड़े शहरों में तो यह समस्या आम इंसान की है ।	8[a]	2	प्रायः सभी व्यक्तियों, अवसरों, अवस्थाओं आदि में पाया जाने वाला या उनसे संबंध रखने वाला
मुझे फलों में आम बड़े पसंद हैं ।	8[a]	4	एक फल जो खाया या चूसा जाता है
पाचनतंत्र दुरुस्त न होने से भोजन अधपचा रहने की समस्या को आम कहते हैं ।	8[a]	2	एक प्रकार का लसदार चिकना सफेद मल जो अन्न न पचने से उत्पन्न होता है

[a] 8 Synset fetched from IndoWordNet for "आम" excluding 1 blank Synset.

Performance of our proposed approach is compared with a method designed by [17] and shown in Table-6 with setup of same tuning parameters.

Table 6. Impact of variation in tuning parameters

Model	Tuning Parameters			WSD Precision	
	Vector Dimension	*Window Size*	*Negative Sampling*	*Previous Approach [17]*	*Proposed Approach*
SkipGram	100	15	0	45.00	49.47
	150	10	5	47.00	51.90
	200	5	10	36.00	41.27
CBOW	100	15	0	41.00	44.90
	150	10	5	52.00	57.21
	200	5	10	46.00	49.44

The next table displays the performance comparison of Individual words.

Table 7. Performance Comparison

Ambiguous Word	Number of Senses		Precision	
	Mentioned and Used in the Previous Approach [17]	*As per Indowordnet & used in our proposed approach*	*Previous Approach [17]*	*Proposed Approach*
आम	4	9*	85.71	91.49
सोना	7	5	14.28	32.18
उत्तर	3	14	28.57	35.40
शाखा	4	6	71.42	71.36
रंग	5	5	100	84.11
मंगल	5	5	57.14	65.78
फल	9	9	85.71	89.5
अंग	5	5	71.42	74.96
दल	12	12	0	24.62

6. Conclusion and future work

This paper proposes a framework to resolve the challenge of ambiguous words using word vectors. The proposed adaptive approach has outperformed the previous approach [17] which is also using word vectors to resolve the ambiguity. The best performance is shown by CBOW model with vector dimensions of 150, window size as 10 and a negative sampling of 5 words. The proposed adaptive approach with the CBOW model and particularly these tuning parameters has shown more than 5% of jump in performance from the previous approach.

The lemmatization of words is not performed before training the model which can further enhance the result. Another challenge is to disambiguate words which are used in idioms. Some of these words have completely different interpretations as to their role in idioms which are completely different from their present context.

References

[1] Ide, Nancy, and Jean Véronis. "Introduction to the special issue on word sense disambiguation: the state of the art." Computational linguistics 24, no. 1 (1998): 1-40.

[2] Agirre, Eneko, and Mark Stevenson. "Knowledge sources for WSD." In Word Sense Disambiguation, pp. 217-251. Springer, Dordrecht, 2007.

[3] Navigli, Roberto. "Word sense disambiguation: A survey." ACM computing surveys (CSUR) 41, no. 2 (2009): 1-69.

[4] Zhong, Zhi, and Hwee Tou Ng. "It makes sense: A wide-coverage word sense disambiguation system for free text." In Proceedings of the ACL 2010 system demonstrations, pp. 78-83. 2010.

[5] Taghipour, Kaveh, and Hwee Tou Ng. "Semi-supervised word sense disambiguation using word embeddings in general and specific domains." In Proceedings of the 2015 conference of the North American chapter of the association for computational linguistics: human language technologies, pp. 314-323. 2015.

[6] Rothe, Sascha, and Hinrich Schütze. "Autoextend: Extending word embeddings to embeddings for synsets and lexemes." arXiv:1507.01127 (2015).

[7] Chen, Xinxiong, Zhiyuan Liu, and Maosong Sun. "A unified model for word sense representation and disambiguation." In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 1025-1035. 2014.

[8] Mahrishi M., Morwal S. (2020) “Index Point Detection and Semantic Indexing of Videos—A Comparative Review”. In: Pant M., Kumar Sharma T., Arya R., Sahana B., Zolfagharinia H. (eds) Soft Computing: Theories and Applications. Advances in Intelligent Systems and Computing, vol 1154. Springer, Singapore.

[9] Mishra, Neetu, Shashi Yadav, and Tanveer J. Siddiqui. "An unsupervised approach to Hindi word sense disambiguation." In Proceedings of the first international conference on intelligent human computer interaction, pp. 327-335. Springer, New Delhi, 2009.

[10] Singh, Satyendr, and Tanveer J. Siddiqui. "Role of semantic relations in Hindi word sense disambiguation." Procedia Computer Science 46 (2015): 240-248.

[11] Agarwal, Madhavi, and Jyoti Bajpai. "Correlation based word sense disambiguation." In 2014 Seventh International Conference on Contemporary Computing (IC3), pp. 382-386. IEEE, 2014.

[12] Patel, Nirali, Bhargesh Patel, Rajvi Parikh, and Brijesh Bhatt. "Hierarchical clustering technique for word sense disambiguation using Hindi WordNet." In 2015 5th Nirma University International Conference on Engineering (NUiCONE), pp. 1-5. IEEE, 2015.

[13] Tayal, Devendra K., Leena Ahuja, and Shreya Chhabra. "Word sense disambiguation in Hindi language using hyperspace analogue to language and fuzzy c-means clustering." In Proceedings of the 12th International Conference on Natural Language Processing, pp. 49-58. 2015.

[14] Singh, Satyendr, and Tanveer J. Siddiqui. "Sense Annotated Hindi Corpus." In 2016 International Conference on Asian Language Processing (IALP), pp. 22-25. IEEE, 2016.

[15] Athaiya, Anidhya, Deepa Modi, and Gunjan Pareek. "A Genetic Algorithm Based Approach for Hindi Word Sense Disambiguation." In 2018 3rd International Conference on Communication and Electronics Systems (ICCES), pp. 11-14. IEEE, 2018.

[16] Sharma, P., and N. Joshi. "Knowledge-Based Method for Word Sense Disambiguation by Using Hindi WordNet." Engineering, Technology & Applied Science Research 9, no. 2 (2019): 3985-3989.
[17] Kumari, Archana, and D. K. Lobiyal. "Word2vec's Distributed Word Representation for Hindi Word Sense Disambiguation." In International Conference on Distributed Computing and Internet Technology, pp. 325-335. Springer, Cham, 2020.
[18] Rumelhart, D.E., Hintont, G.E., Williams, R.J.: Learning representations by backpropagating errors. Nature 323, 533–536 (1986)
[19] Bengio, Y., Schwenk, H., Senécal, J.S., Morin, F., Gauvain, J.L.: Neural probabilistic language models. In: Holmes, D.E., Jain, L.C. (eds.) Innovations in Machine Learning. STUDFUZZ, vol. 194, pp. 137–186. Springer, Heidelberg. 2006.
[20] R. Collobert and J. Weston: A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning. Architecture, vol. 20, no. 1, pp. 160 – 167, 2008
[21] T. Mikolov, G. Corrado, K. Chen, and J. Dean: Efficient Estimation of Word Representations in Vector Space. Proc. Int. Conf. Learn. Represent. (ICLR 2013), pp. 112, 2013
[22] T. Mikolov, K. Chen, G. Corrado, and J. Dean: Distributed Representations of Words and Phrases and their Compositionality. Nips, pp. 1 – 9, 2013
[23] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov: Bag of tricks for efficient text classification. arXiv Prepr. arXiv1607.01759 (2016)
[24] Soni, Vimal Kumar, Dinesh Gopalani, and M. C. Govil. "Resurgence of Deep Learning: Genesis of Word Embedding." In Smart Innovations in Communication and Computational Sciences, pp. 129-139. Springer, Singapore, 2019.
[25] Siddiqi, Sifatullah, and Aditi Sharan. "Construction of a generic stopwords list for Hindi language without corpus statistics." International Journal of Advanced Computer Research 8, no. 34 (2018): 35-40.