Word Sense Disambiguation

Tanishq Goel
Aaryan Singh
Veeral Agarwal

CL-2 Project Report

Prof. Dipti Sharma & Pruthwik Mishra

Word Sense Disambiguation

Abstract

Several studies have tried to improve retrieval performances based on automatic Word Sense Disambiguation techniques. So far, most attempts have failed. We try, through this paper, to give a deep analysis of the reasons behind these failures.

Our aim through this paper is not to propose sophisticated strategies to improve retrieval performances using a word sense disambiguation (WSD) algorithm. Rather we mainly want to explore whether WSD can be useful in Information Retrieval (IR).We deeply analyzed the obtained results and formulated some conclusions and perspectives.

Introduction

We understand that words have different meanings based on the context of its usage in the sentence. If we talk about human languages, then they are ambiguous too because many words can be interpreted in multiple ways depending upon the context of their occurrence.

Therefore, WSD becomes a very crucial component in MT systems. If the sense of a word is not correctly identified, the translation can become jarring. It was basically a task for us to identify the correct meaning of a word in the given context. As a basic semantic understanding at the lexical level, WSD is a fundamental problem in NLP.

Determining which “sense” of a word is activated by the use of the word in a particular context, which can be a phrase, a sentence or a paragraph. So, WSD is basically a task of finding the correct sense of the words and automatically assigning its correct sense to the words which may be polysemous in some cases. We can easily find out the correct meaning of a word in context, but for machines it’s a difficult task.

For example, consider the two examples of the distinct sense that exist for the word “plant” −

- This **plant** is will gloom into a tree.
- Jamshedpur is famous for its steel **plants**.

The occurrence of the word **plant** clearly denotes the distinct meaning. In first sentence, it means a living organism of the kind exemplified by trees, shrubs, herbs, grasses etc. and in second, it means a factory. Hence, if it would be disambiguated by WSD then the correct meaning to the above sentences can be assigned as follows −

- This **plant/herb** is will gloom into a tree.
- Jamshedpur is famous for its steel **plants/factories**.

Evaluation of WSD

The evaluation of WSD requires the following two inputs −

A Dictionary

The very first input for evaluation of WSD is dictionary, which is used to specify the senses to be disambiguated.

Test Corpus

Another input required by WSD is the high-annotated test corpus that has the target or correct-senses. The test corpora can be of two types;

- Lexical sample − This kind of corpora is used in the system, where it is required to disambiguate a small sample of words.
- All-words − This kind of corpora is used in the system, where it is expected to disambiguate all the words in a piece of running text.

Scope

We aim to use Indo Wordnet for finding out the senses used in source sentences and evaluate the performance of different MT systems.

1. We extracted 100 Hindi sentences from Indo WordNet where sentences were centered around words having multiple senses.

2. We Translated them to English, Telugu, Punjabi using both Bing and Google.

3. We validated all of the 100 sentences with the help of our friends having their linguistic background as Punjabi or Telugu.

4. We started developing a WSD system for Indian Languages
   - Lesk Algorithm
   - Word Embeddings

5. We deeply analyzed the obtained results and formulated some conclusions and perspectives.

Survey

We also conducted a survey about challenges faced by people in MT due to word sense disambiguation failures.

Link to the survey [click here] (can also be found in the report directory)

Detailed Summary [click here] (can also be found in the report directory)

Data Collection, Processing and Annotation

- Since, our task is about WSD and then implementing them into MT using Indo-Wordnet, we initially picked up 100 Hindi sentences from the Hindi Wordnet, where sentences were centered around words having multiple senses.

- Then these sentences were looked upon to check whether we have got many clusters of sentences in which a word is common and have different senses.

- Then, the data was manually annotated with their Synset IDs so as to check against the output from our algorithmic analysis and such.

- After we had chosen those initial 100 sentences, we translated them to English, Telugu and Punjabi [using Google Translate and Bing Translate]

- Since, all three of us involved in the project only know Hindi and English, we validated the authenticity of the translations in other languages with the help of our classmates. For that we would like to thank Konda Jayant Reddy, P Sahithi Reddy for their valuable inputs in Telugu and Pratham Gupta, Dinesh Garg, Ashish Gupta for Punjabi.

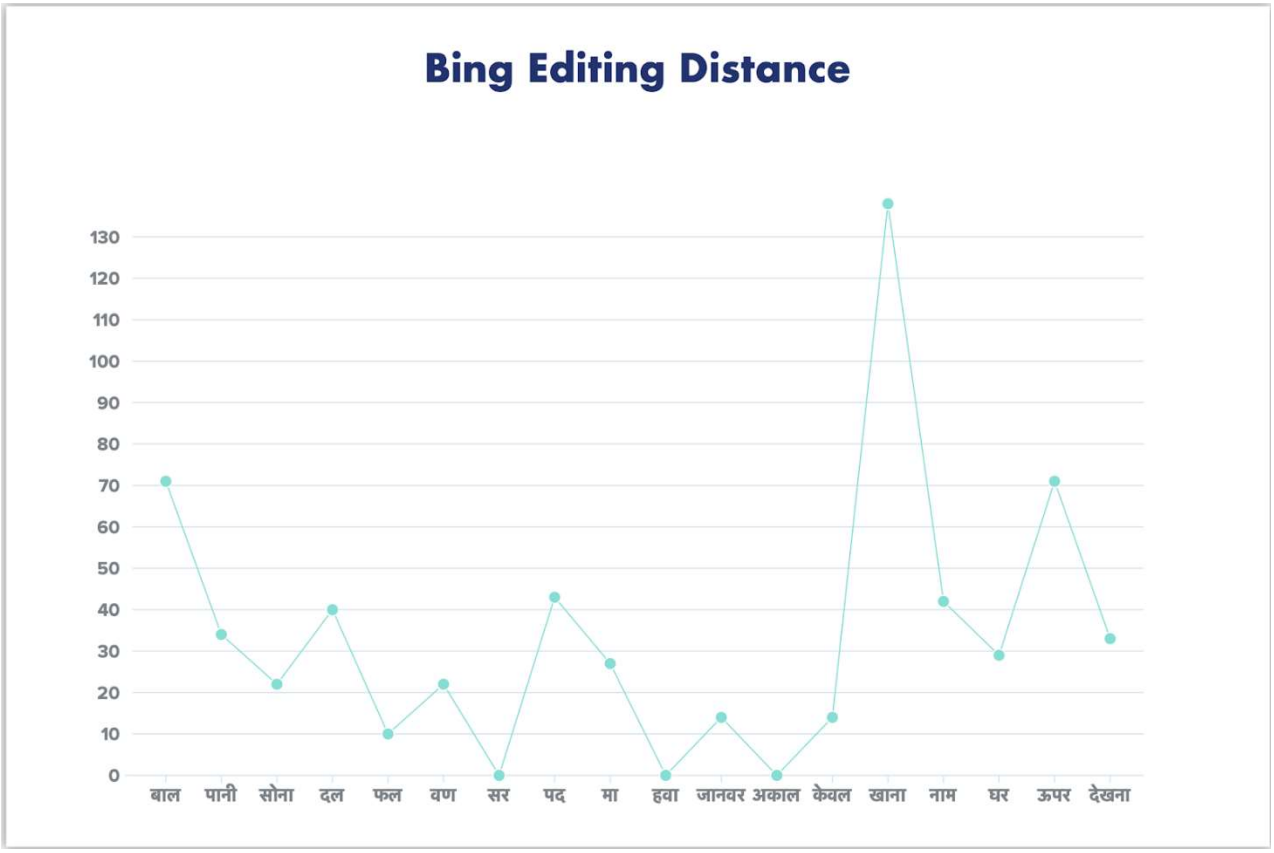
- They marked those sentences whose translation had some minor inconsistencies, marked another set of sentences which differed much in their senses differently, for both sets of translations, I.e. the Bing translation and Google Translation.

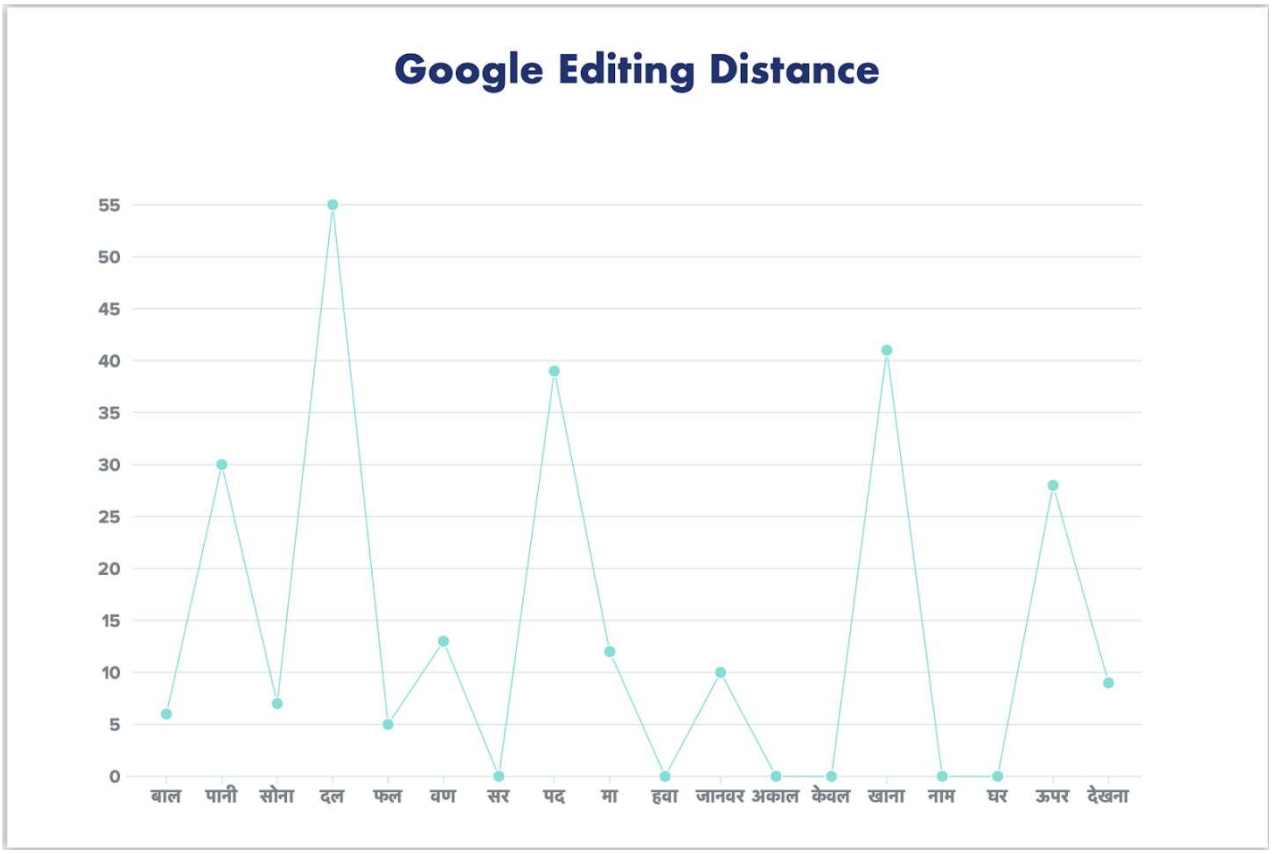
- Then, we used Editing Distance as a parameter for measuring the degree of deviations from what their original translations should be.

Major Insights

- Translation in itself is a very tedious job, since you have to find words nearest to the meaning which the original sentence is trying to convey, because true translation almost never holds.

- Our Hindi sentences were comprised with sentences having different senses of some words, which are : बाल, पानी, सोना, दल, फल, वण, सर, पद, मा, हवा, जानवर, अकाल, केवल, खाना, नाम, घर, ऊपर, देखना .In these sentences, the sentences having different senses of देखना were captured best by Google and Bing Translations, whereas sentences having different senses of खाना were found to be most inconsistent.





- The above graphs map Editing distance versus the various senses which comprised our 103 sentences. It clearly shows that while for some specific words, Bing performs better translation, but overall Google translation is better.

- Google or Bing translation provided the best results, i.e. fewest inconsistencies, for Hindi-English pair, followed by Hindi-Punjabi pair, which was followed by Hindi-Telugu pair.

- Bing Translation is inferior when compared to Google Translate, and it holds for all the language, i.e. Hindi, Telugu and Punjabi.

- Total Editing Distance between the 100 Google Translated sentences and the authentic translated sentences for Hindi-English pair was found to be 255, which were spread across 19 sentences, which were having inconsistencies. In contrast, the

total Editing Distance between Bing Translated sentences and the authentic ones for the same pair was found to be 610, spreading across 45 wrongly translated sentences. Attached are the individual files mentioning Edit distances from both Bing and Google.

- Furthermore, the distribution among the incorrect sentences was not even, some differed only slightly, while some differed greatly. Sometimes, due to the relative free word-ordering in Hindi, their translation made no sense, and we had to rewrite the translated sentences in totality.

Lesk Algorithm

The Lesk algorithm is based on the assumption that words in a given "neighborhood" (section of text) will tend to share a common topic. A simplified version of the Lesk algorithm is to compare the dictionary definition of an ambiguous word with the terms contained in its neighborhood.

1. for every sense of the word being disambiguated one should count the amount of words that are in both neighborhood of that word and in the dictionary definition of that sense
2. the sense that is to be chosen is the sense which has the biggest number of this count

In Simplified Lesk algorithm, the correct meaning of each word in a given context is determined individually by locating the sense that overlaps the most between its dictionary definition and the given context. Rather than simultaneously determining the meanings of all words in a given context, this approach tackles each word individually, independent of the meaning of the other words occurring in the same context.

It is based on the hypothesis that words used together in text are related to each other and that the relation can be observed in the definitions of the words and their senses. Two (or more) words are disambiguated by finding the pair of dictionary senses with the greatest word overlap in their dictionary definitions. It searches for the shortest path between two words: the second word is iteratively searched among the definitions of every semantic variant of the first word, then among the definitions of every semantic variant of each word in the previous definitions and so on. Finally, the first word is disambiguated by selecting the semantic variant which minimizes the distance from the first to the second word.

We updated Lesk Algorithm for English and then implemented Lesk algorithm for Indian Languages using Pyiwn.

Pyiwn: A Python-based API to access Indian Language Wordnets. This API gives access to synsets, glosses, examples, lexico-semantic relations between synsets, ontology nodes for 18 Indian languages, viz., Assamese, Bangla, Bodo, Gujarati, Hindi, Kannada, Kashmiri, Konkani, Malayalam, Meitei (Manipuri), Marathi, Nepali, Odia, Punjabi, Sanskrit, Tamil, Telugu and Urdu. In future, it will also provide access to speech data for words, glosses examples in Hindi WordNet.

You can find the link to codes here:
Lesk for English (Link) (can be found in codes section)
Lesk for Hindi (Link) (can be found in codes section)

Following are some of the outputs of the above code shown here as an example:

[Input]

बाल "नाई की दुकान में जगह-जगह बाल दिखाई दे रहे थे"
[Output]
Result is: बालों का समूह

[Input]
बाल "कीटनाशक का छिड़काव न करने से अनाज की बालों में कीड़े लग गए हैं"
[Output]
Result is: गेहूँ, ज्वार, बाजरे आदि के पौधों का वह अगला भाग जिस पर दाने होते हैं

Word Embeddings

As the Wikipedia will point out, word embedding is

'the collective name for a set of language modeling and feature learning techniques in natural language processing (NLP) where words or phrases from the vocabulary are mapped to vectors of real numbers.'

The basic idea of word embedding is words that occur in similar context tend to be closer to each other in vector space. For generating word vectors in Python, modules needed are **`nltk`** and **`gensim`**.

Below is the implementation:

```
# Python program to generate word vectors using Word2Vec

# importing all necessary modules
from nltk.tokenize import sent_tokenize, word_tokenize
import warnings

warnings.filterwarnings(action = 'ignore')

import gensim
from gensim.models import Word2Vec

#  Reads 'alice.txt' file
sample = open("C:\\Users\\Admin\\Desktop\\alice.txt", "r")
s = sample.read()

# Replaces escape character with space
f = s.replace("\n", " ")

data = []

# iterate through each sentence in the file
```

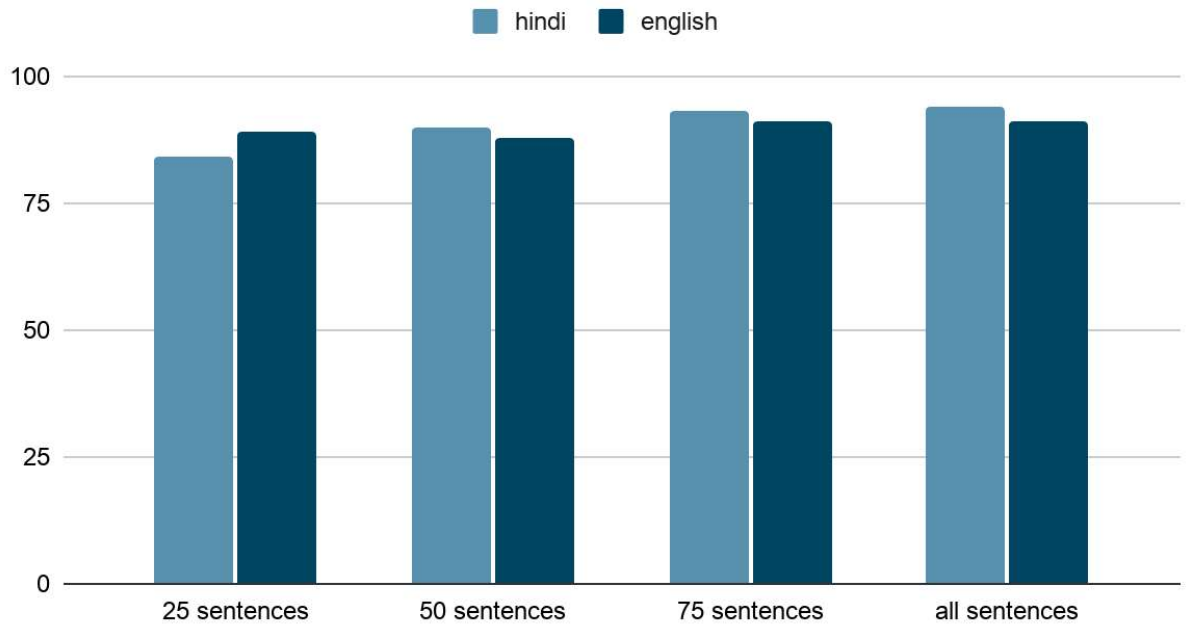
```
for i in sent_tokenize(f):
    temp = []

    # tokenize the sentence into words
    for j in word_tokenize(i):
        temp.append(j.lower())

    data.append(temp)

# Create CBOW model
model1 = gensim.models.Word2Vec(data, min_count = 1,
                                size = 100, window = 5)

# Print results
print("Cosine similarity between 'alice' " +
               "and 'wonderland' - CBOW : ",
    model1.similarity('alice', 'wonderland'))

print("Cosine similarity between 'alice' " +
                 "and 'machines' - CBOW : ",
      model1.similarity('alice', 'machines'))

# Create Skip Gram model
model2 = gensim.models.Word2Vec(data, min_count = 1, size = 100,
                                             window = 5, sg = 1)

# Print results
print("Cosine similarity between 'alice' " +
          "and 'wonderland' - Skip Gram : ",
    model2.similarity('alice', 'wonderland'))

print("Cosine similarity between 'alice' " +
            "and 'machines' - Skip Gram : ",
      model2.similarity('alice', 'machines'))
```

Output :

```
Cosine similarity between 'alice' and 'wonderland' - CBOW :  0.999249298413
Cosine similarity between 'alice' and 'machines' - CBOW :  0.974911910445
Cosine similarity between 'alice' and 'wonderland' - Skip Gram :  0.885471373104
Cosine similarity between 'alice' and 'machines' - Skip Gram :  0.856892599521
```

Observations

- Major errors are occurring in very ambiguous senses, otherwise working fine.
    - For example consider “After visiting the bank he went to swim in the river nearby”
    - Here bank can have two possible senses but lesk algorithm works on the basis of neighboring words so here output will be sense of river bank which can be correct or incorrect depending on the context. These minor inaccuracies leads to significant errors.



- From the above point we observe that accuracy of lesk is staying more or less same with changes in the dataset. From a point of view of whole Hindi Wordnet all the possible cases seem small so we can increase the dataset. Scope of further improvement is possible in our Hindi dataset containing synset annotated sentences.

Drawbacks and Challenges Faced

- There is never a true one on one translation for words with not so common senses.
- Recognizing the sense is very tough, even manually. Often the senses annotated individually by us showed different Synset Ids when crossed with the machine output.
- When we computed Editing distance, so as to compare the deviations in Google and Bing translations, we found out that the results had been affected by how the reference sentences were set, i.e. whether we corrected the sentences from their Google translated versions or from the Bing ones. Here, we corrected the Google translated sentences and set them as our references.
- However, to overcome this phenomenon, we tried to manually manipulate editing distance where the translations were different but pointed to the same senses. For eg. Google translated a sentence as : the working of the machine whereas, Bing translated the same as : the functioning of the instrument. Here, the machine will show the editing distance to be quite significant between these two sentences, but in reality they are almost the same. Hence, special caution has been given to such sentences.
- Leak algorithm can give only one output i.e. most related according to neighbouring words. But there can be more than one senses that can't be identified.
- Due to less time we were not able to complete the model of WSD using word embeddings otherwise we could have compared to get a better understanding.

Conclusion

- Google or Bing translation provided the best results, i.e. fewest inconsistencies, for Hindi-English pair, followed by Hindi-Punjabi pair, which was followed by Hindi-Telugu pair.

- Bing Translation is inferior when compared to Google Translate, and it holds for all the language, i.e. Hindi, Telugu and Punjabi.

- Total Editing Distance between the 100 Google Translated sentences and the authentic translated sentences for Hindi-English pair was found to be 255, which were spread across 19 sentences, which were having inconsistencies. In contrast, the total Editing Distance between Bing Translated sentences and the authentic ones for the same pair was found to be 610, spreading across 45 wrongly translated sentences. Attached are the individual files mentioning Edit distances from both Bing and Google.

- Furthermore, the distribution among the incorrect sentences was not even, some differed only slightly, while some differed greatly. Sometimes, due to the relative free word-ordering in Hindi, their translation made no sense, and we had to rewrite the translated sentences in totality.

- "A comparative evaluation performed by Vasilescu et al. (2004)[4] has shown that the simplified Lesk algorithm can significantly outperform the original definition of the algorithm, both in terms of precision and efficiency. By evaluating the disambiguation algorithms on the Senseval-2 English all words data, they measure a 58% precision using the simplified Lesk algorithm compared to the only 42% under the original algorithm.

- If the target concept was found in the synsets, then we marked the sense as captured in the translation, else the translation was marked incorrect. In this way by evaluating lesk algorithm over our dataset we reached astonishingly, 95% accuracy which is a huge number to begin with. We feel that as dataset increases , more and more exceptions start coming and would lead to a decrease in this accuracy.

References

- https://github.com/riteshpanjwani/pyiwn
- https://www.c-sharpcorner.com/article/lesk-algorithm-in-python-to-remove-word-ambiguity/
- https://en.wikipedia.org/wiki/Lesk_algorithm
- http://compling.hss.ntu.edu.sg/events/2018-gwc/presentations/GWC2018_40.pdf
- https://package.wiki/pyiwn
- https://www.semanticscholar.org/paper/pyiwn-%3A-A-Python-based-API-to-access-Indian-Panjwani-Kanojia/b6de7f85baae6e1795ce89dbefe8430c1f6e6855
- https://www.geeksforgeeks.org/python-word-embedding-using-word2vec/