Proceedings of the WILDRE-6 Workshop @LREC2020, pages 18–23
Marseille, 20 June 2022

HindiWSD: A Package for Word Sense Disambiguation in Hinglish & Hindi

Mirza Yusuf, Praatibh Surana, Chethan Sharma*
Department of Information and Communication Technology, Manipal Institute of Technology, Manipal Academy of Higher Education, India
mirzayusuf1000@gmail.com, praatibhsurana@gmail.com, chethan.sharma@manipal.edu

Abstract

A lot of commendable work has been done, especially in high resource languages such as English, Spanish, French, etc. However, work done for Indic languages such as Hindi, Tamil, Telugu, etc is relatively less due to difficulty in finding relevant datasets, and the complexity of these languages. With the advent of IndoWordnet, we can explore important tasks such as word sense disambiguation, word similarity, and cross-lingual information retrieval, and carry out effective research regarding the same. In this paper, we worked on improving word sense disambiguation for 20 of the most common ambiguous Hindi words by making use of knowledge-based methods. We also came up with "hindiwsd", an easy- to-use framework developed in Python that acts as a pipeline for transliteration of Hinglish code-mixed text followed by spell correction, POS tagging, and word sense disambiguation of Hindi text. We also curated a dataset of these 20 most used ambiguous Hindi words. This dataset was then used to enhance a modified Lesk algorithm and more accurately carry out word sense disambiguation. We achieved an accuracy of about 71% using our customized Lesk algorithm which was an improvement to the accuracy of about 34% using the original Lesk algorithm on the test set.

Keywords: Word Sense Disambiguation, Code-Mixing, Indic Transliteration

1. Introduction

The use of a Hindi-English mix language usually referred to as Hinglish has been used prominently since the inception of social media. According to a recent survey, around 57% of the Indian population generally while conversing on any social media prefer to use Hinglish over Devanagari Hindi or English[1]. The popularity of Hinglish arises from the fact that it is easier than typing in Hindi due to the unavailability of Hindi characters on a common keyboard. Wordnets (George A. Miller. 1995) are used extensively for many NLP-related tasks. They are generally used for a number of processes in information systems, including word sense disambiguation (which from here on we will refer to as WSD), information retrieval, automatic text classification, automatic text summarization, machine translation, and even automatic crossword puzzle generation[2]. The Indian languages wordnet originated from the advent of the Hindi Wordnet (Bhattacharyya et al., 2008). Using this model as default, the wordnets for other Indic languages were developed. Eighteen of these languages have wordnets under a common platform known as the IndoWordNet (Pushpak Bhattacharyya, 2010). In Natural Language Processing, WSD is the problem of determining which sense of a word is being used in a particular sentence. Given a word, it can have multiple possible meanings which makes it difficult for a reader/system to understand the meaning of the word by itself. However, when the word is provided along with the context/sentence it is a part of, it becomes easier to gauge its meaning. The process of identifying this contextual meaning of the word is called Word Sense Disambiguation". For
example, the Hinglish sentence Mein sonaa chaahta hoon translates as I want to sleep, whereas in the sentence Mein sonaa khareednaa chaahta hoon translates as I want to buy gold, here the word sonaa is being used in two different contexts and hence has two different meanings; the first one being sleep and the other being gold. The presence of just 1 extra word was able to change the meaning of the word sonaa. Hence, one can see why it would be tough for a model to carry out WSD.

2. Related Work

WordNet is a large lexical database of English. Nouns, verbs, adjectives, and adverbs are grouped into sets of cognitive synonyms (synsets), each expressing a distinct concept. Synsets are interlinked by means of conceptual-semantic and lexical relations. WordNet is also freely and publicly available for download. Its structure makes it a useful tool for computational linguistics and natural language processing. In recent times, it has been developed in other languages as well such as French, German, and in our case Hindi as can be seen with the Hindi WordNet. Word Sense Disambiguation has been done for English by utilizing

[1] https://www.milestoneloc.com/guide-to-hinglish-language/

[2] https://en.wikipedia.org/w/index.php?title=WordNet&oldid=1069690711

the data from the original Wordnet, which contains a vast amount of relations between words. Various methods, both supervised, and unsupervised have been used. Apart from this, Banerjee and Pedersen (2002) also made use of knowledge-based methods such as the Lesk algorithm (Michael Lesk, 1986). Seo et al. (2004) made use of unsupervised methods on English data and then evaluated their model on Korean datasets. Mihalcea and Faruque (2004) made use of a supervised approach with a minimum amount of annotated data and achieved commendable results on SENSEVAL-3 (Snyder, Benjamin and Palmer, 2004) in all English tasks. Supervised and semi-supervised techniques have tended to outperform the knowledge-based methods as they use machine learning, which to an extent can identify the semantic structure of a sentence which knowledge-based methods fail to. Pal and Saha (2015) also suggest that while precision is high for knowledge-based approaches, supervised approaches are best for languages with rich amounts of data. In the case of Hindi and Hinglish code mixed data, however, supervised approaches will only do so well due to some level of data scarcity. This prompted us to employ knowledge-based methods for our specific problem statement. With regards to Hindi, Singh et al. (2013) carried out WSD by computing similarity based on semantics. They were able to achieve a commendable accuracy of about 60% on 20 polysemous Hindi nouns. Sinha et al. (2004) also carried out a detailed evaluation of ambiguous Hindi words and evaluated accuracies of particular words grouped on the basis of domains. Gautam and Sharma (2016) used an interesting approach wherein they made use of bigrams and trigrams to disambiguate 15 commonly used Verb Hindi words. They achieved the highest precision of about 53% on bigram words. All these papers leveraged the Hindi WordNet. Through our model, we have tried to focus on Hinglish code mixed as well as Hindi data. In the case of WSD for Hinglish code mixed data, the only extra step is the transliteration of this Hinglish data to Devanagari Hindi post which we carry on with our usual algorithm as is described in the methodology section.

3. Methodology

A pipeline was created to ensure our model could handle both Hinglish code-mixed as well as Hindi Devanagari data. Apart from carrying out WSD on Devanagari Hindi, we also leveraged pre-existing tools for spell correction, POS tagging, etc. The following sections describe the algorithm used for WSD and modifications made to it to further improve accuracy.

3.1. Lesk Algorithm

Lesk algorithm is a knowledge-based algorithm for WSD. It is based on the assumption that words in a given neighborhood will tend to share the same topic. Simplified lesk algorithm is used to compare the dictionary definition of an ambiguous word with the terms contained in its neighborhood and take an overlap of them. The highest overlap is then processed as the correct meaning. The pseudo-code is shown in algorithm 1 along with the flow diagram in Figure 1 and Figure2.

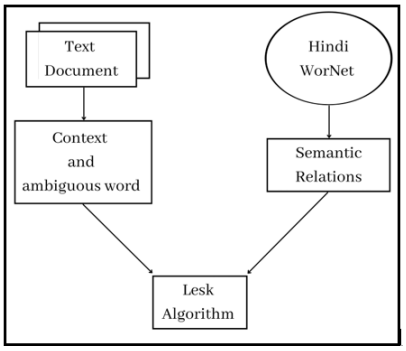

Figure 1: Traditional Lesk algorithm

```
function LESK (word, sentence) returns best sense of word
        best sense <- most frequent use for word
        max_overlap <- 0
        context <- set of words in sentence
        for each sense in sense of word do
                signature <- set of words in the gloss and examples of sense
                overlap <- COMPUTEOVERLAP (signature, context)
                if overlap > max_overlap then
                        max_overlap <- overlap
                        best_sense <- sense
end return (best_sense)
```

Figure 2: Traditional Lesk algorithm

3.2. Custom Lesk Algorithm

In addition to the traditional Lesk algorithm, we employed a modified Lesk algorithm that made use of a helper dataset, tailored specifically for our task of disambiguating the most commonly used Hindi words. An intersection was taken with the input sentence and the keywords in the helper dataset to find which meaning for a particular word had the highest overlap. This was then compared to the initial lesk overlap, the highest overlap was then chosen as the meaning of the ambiguous word. As opposed to the Lesk algorithm that required

the POS tags[3] for words to be disambiguated, our custom algorithm leverages the synsets present in the Hindi WordNet to predict the meanings. The pseudo-code is as follows along with the flow diagram in Figure 4.

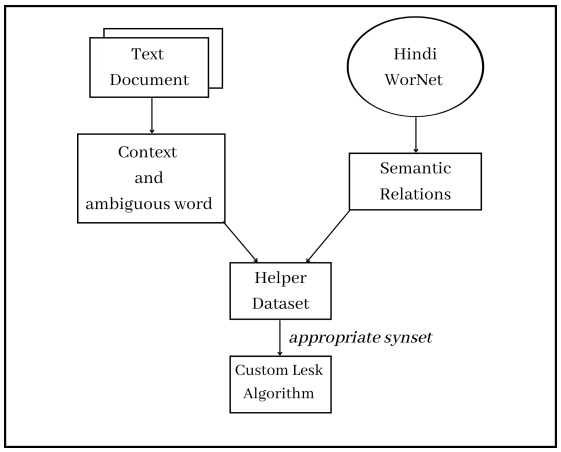

Figure 3: Custom Lesk algorithm

```
function LESK (word, sentence) returns best sense of word
        best sense <- most frequent use for word
        max_overlap <- 0
        context <- set of words in sentence
        intersection, synset_no, meaning =
EXTRAOVERLAP (word, sentence)
        for each sense in sense of word do
                signature <- set of words in the gloss
and examples of sense
                overlap <- COMPUTEOVERLAP
                (signature, context)
                if overlap > max_overlap then
                        max_overlap <- overlap

        if intersection > max_overlap and
intersection > 0 do
                best_sense <- word.synset_no
end return (best_sense)
```

Figure 4: Custom Lesk algorithm

3.3. Library Design and Features

The hindiwsd package is essentially a pipeline that will carry out the following tasks- Hinglish to Hindi transliteration Spell correction of Hindi text POS tagging of Hindi text Word Sense Disambiguation of Hindi text with the help of IndoWordNet Enhanced disambiguation using custom Lesk algorithm and custom dataset

A user can give input either as a Hinglish sentence or as a Hindi sentence to the wordsense function (see snippet 3.3). The wordsense function prints out each word of the sentence along with its disambiguated word meaning if the respective Hindi word is present in the IndoWordNet.

```
>>> from hindiwsd import hindi_wsd
>>> print(hindi_wsd.wordsense("aaj achha din hai"))  #Hinglish
>>> print (hindi_wsd.wordsense("आज अच्छा दिन है"))  #Hindi
```

3.3.1. Library Design

The hindiwsd package is built on top of a few scripts taken from other pre-existing libraries with modifications made to them to make them compatible with hindiwsd and modules written by the authors. With the help of pre-existing libraries, we were able to ensure the dependable functionality of the pipeline as a whole and provide a well-compiled multi-faceted model to be used for Hindi and Hinglish WSD. The pipeline along with the working of the key features has been explained in the Features section.

3.3.2. Features

The input can either be a Hindi sentence in its Devanagari form or a Hinglish code mixed sentence. In any case, the sentence is then converted to Devanagari Hindi using the indic-transliteration tool[4]. After this, spell correction is performed using Spello[5]. The resultant sentence is then passed through a POS tagger and the tags are then condensed and converted into 4 simple tags, i.e., noun, adjective, verb, and adverb. This is done in order to be able to use pyiwn (Panjwani et al., 2018) and access the IndoWordNets synsets. Figure 5 shows the pipeline that the data flows through.

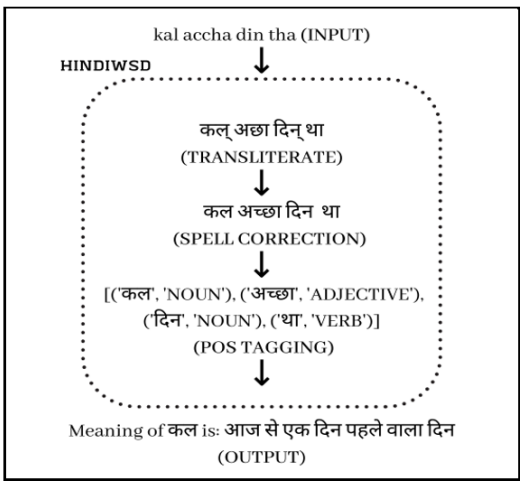

Figure 5: Pipeline for Hindi WSD

[3] https://nptel.ac.in/courses/106101007

[4] https://github.com/indic-transliteration/indic_transliteration_py

[5] https://github.com/hellohaptik/spello

3.3.3. Hinglish to Hindi Transliteration using indic-transliteration

The indic-transliteration package helps convert the Hinglish code-mixed data to Hindi by making use of dictionaries along with rules specified for the conversion of each set of letters in a word from Latin script to Devanagari Hindi.

3.3.4. Spell correction using spello

The spello package makes use of two spell correction models, namely, Phoneme which uses the Soundex algorithm[6], and Symspell[7]. The Phoneme model makes use of the Soundex algorithm and suggests spellings based on phonetics. The Symspell model uses edit distances to suggest spell corrections. Spello combines both these models and provides accurate spell correction.

3.3.5. POS tagging with indic_tagger

The indic_tagger package[8] is a state-of-the-art POS tagger and chunker for Indian languages. We made use of the pre-trained CRF model for chunking[9] and POS tagging. In order to make the tags compatible and useful for the Lesk algorithm, we carried out a mapping of the numerous model generated tags to 4 tags, i.e., noun, adjective, verb, and adverb to make the tags useful for Lesk algorithm. The POS mappings are shown in Figure 6

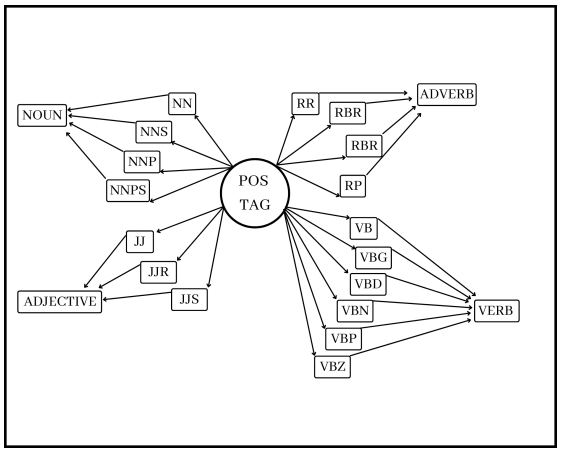

Figure 6: POS Mappings

3.3.6. Improved word sense disambiguation with pyiwn

The pyiwn package is an API developed to access the IndoWordNet. Using pyiwn, we were able to access the synsets[10], glosses, examples, and lexico-semantic relations[11] between synsets. We were able to perform the Lesk algorithm on our data successfully. Apart from this, as mentioned earlier, a helper dataset was developed for the most commonly used Hindi words to further aid our custom Lesk algorithm and make WSD more robust.

4. Helper Dataset

The Helper dataset is a small dataset that was manually curated by us for the purpose of enhancing Lesk algorithm. It contains 20 of the more commonly used ambiguous Hindi words and words which occur frequently with those particular words. Due to time constraints and limited manpower, we curated the dataset to only hold the two most frequent meanings of an ambiguous word despite there being more. The dataset was structured in a way such that for each meaning of a word, we provide two sets of keywords generated from random sentences consisting of the ambiguous word. See 7 for a better understanding. In the example in

Word	Sentence
पता	मेरा नाम नहीं लगेगा सब बारे उसके घटना चलती चला है मुझे उसे हमें चले यह लगाने कैसे चलता लगाते लगा
पता	क्या जगह घर दफ्तर पते पहुंच वहां जाना जा मालुम पूछना

Figure 7: Helper dataset example

table 1, we try to showcase the meaning of the word पता in different sentences where the context is different. In the first row the word means got to know, whereas, in the second row, it means address. To a native speaker, this is rather straightforward to identify with the help of the context, that is, the surrounding or supporting words in the sentence.

5. Results

Table 2 represents the results that we obtained when we tested a word for a particular meaning. The synset column in the table signifies the particular meaning of the word as indicated in the IndoWordNet. The test set contains sentences for each meaning along with the ambiguous word in question. Along with the synset number, the meaning corresponding to that particular synset is shown in the meaning column. Finally, the results of the traditional and modified Lesk algorithms are displayed.

Figure 8 shows words and their respective meanings along with the results obtained for both Lesk

[6] https://en.wikipedia.org/w/index.php?title=Soundex&oldid=1055316589

[7] https://github.com/wolfgarbe/SymSpell

[8] https://github.com/avineshpvs/indic_tagger

[9] https://www.analyticsvidhya.com/blog/2021/10/what-is-chunking-in-natural-language-processing/

[10] https://www.nltk.org/howto/wordnet.html

[11] https://en.wikipedia.org/w/index.php?title=Lexical_semantics&oldid=1041088037

Word	Synset	Meaning	Lesk	Modified Lesk
कल	2	आज के बाद आने वाला पहला दिन	1	1
कल	0	आज से एक दिन पहले वाला दिन	1	1
सोना	2	लेटकर शरीर और मस्तिष्क को विश्राम देने वाली निद्रा की अवस्था में होना	0	1
सोना	0	एक बहुमूल्य पीली धातु जिसके गहने आदि बनते हैं	1	1
कर	2	वह नियत धन आदि जो किसी व्यक्ति या किसी संपत्ति, व्यापार आदि के काम में से कोई अधिकारिकी अपने लिए लेती है	0	1
कर	N	हाथ	0	1
चाल	4	कामयाबी पाने के लिए चालाकीपूर्वक लगाई जाने वाली युक्ति	0	0
चाल	2	व्यवहार की वह प्रकृति जो लगातार दोहराव से प्राप्त होती है	1	1

Figure 8: Results for a few common Hindi words and their meanings

algorithms that were used. The comparison makes it evident that our modified Lesk algorithm outperforms the normal Lesk algorithm. This was what we expected as our helper dataset was tailored to help improve disambiguation for commonly used Hindi words.

Accuracy = Correctly predicted samples / Total predicted samples (1) Table 3 shows the final results measured using accuracy as the performance metric This also shows that instead of maybe approaching WSD as a broad task, one can break it down into smaller, more specific, and meaningful subtasks, allowing for better accuracy and greater utility.

	Lesk	Modified Lesk
Accuracy (2 meanings of 20 most common Hindi words)	34.21%	71.05%

6. Conclusion and Further Work

Creating a WSD model for a language with a scarce amount of resources is a cumbersome task. In this paper, we have tried our best to limit our problem statement and focus on a task with real-world applications, hence we chose to go with common words and dictionary-based methods while also coming up with utilities that can aid further research in this domain. The Lesk algorithm is not the most efficient way to perform WSD since it is incapable of understanding context or even the semantics involved. However, this algorithm is easy to use, quick to make predictions and can be used to target specific subtasks even within WSD as was demonstrated in this paper. Due to limited resources, we did not feel it would be the best approach to employ supervised methods. Further work can involve increasing the size of the dataset and maybe involving machine translation systems (Appicharla et al., 2021).

7. Bibliographical References

Appicharla Ramakrishna & Gupta, Kamal & Ekbal, Asif & Bhattacharyya, Pushpak, (2021). IITP-MT at CALCS2021: English to Hinglish Neural Machine Translation using Unsupervised Synthetic Code-Mixed Parallel Corpus. 31-35. 10.18653/v1/2021.calcs-1.5.

Banerjee, Satanjeev & Pedersen, Ted. (2002). An Adapted Lesk Algorithm for Word Sense Disambiguation Using WordNet. Computational Linguistics and Intelligent Text Processing. 2276. 136-145. 10.1007/3-540-45715-1_11.

Chandra Bhal Singh Gautam and Dilip Kumar Sharma. (2016). Hindi Word Sense Disambiguation Using Lesk Approach on Bigram and Trigram Words. *Proceedings of the International Conference on Advances in Information Communication Technology & Computing (AICTC '16)*. In Association for Computing Machinery, New York, NY, USA, Article 81, 1–5.

Michael Lesk. (1986). Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone. *In Proceedings of the 5th annual international conference on Systems documentation (SIGDOC '86)*. Association for Computing Machinery, New York, NY, USA, 24–26.

Mihalcea, Rada, 1974- & Faruque, Ehsanul. SenseLearner: Minimally Supervised Word Sense Disambiguation for All Words in Open Text, paper, (2004); [Stroudsburg, Pennsylvania], University of North Texas Libraries, UNT Digital Library.

Pal, Alok Ranjan and Diganta Saha.(2015). "Word sense disambiguation: a survey." ArXiv abs/1508.01346: n. pag

8. Language Resource References

Bhattacharyya, Pushpak, Prabhakar Pande, and Laxmi Lupu. (2008). Hindi WordNet LDC2008L02. Web Download. Philadelphia: Linguistic Data Consortium.

George A. Miller. (1995). WordNet: a lexical database for English. Commun. ACM 38, 11 (Nov. 1995), 39–41.

Pushpak Bhattacharyya. 2010. IndoWordNet *In Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10).*