COMPUTATIONAL LINGUISTICS - 1
SPRING SEMESTER 2021

INSTRUCTOR: RADHIKA MAMIDI

TA : MOUNIKA MAREDDY

Team : Shubhankar Kamthankar - 2020114004
Yash Agrawal - 2020114005
Prayush Rathore -2020114009

Topic: Sentiment Analysis for the Hindi Language - A Rule-Based approach

---

Abstract

In this report, we are describing the working and analysis on a Sentiment analyzer for the Hindi Language.

Resources used :

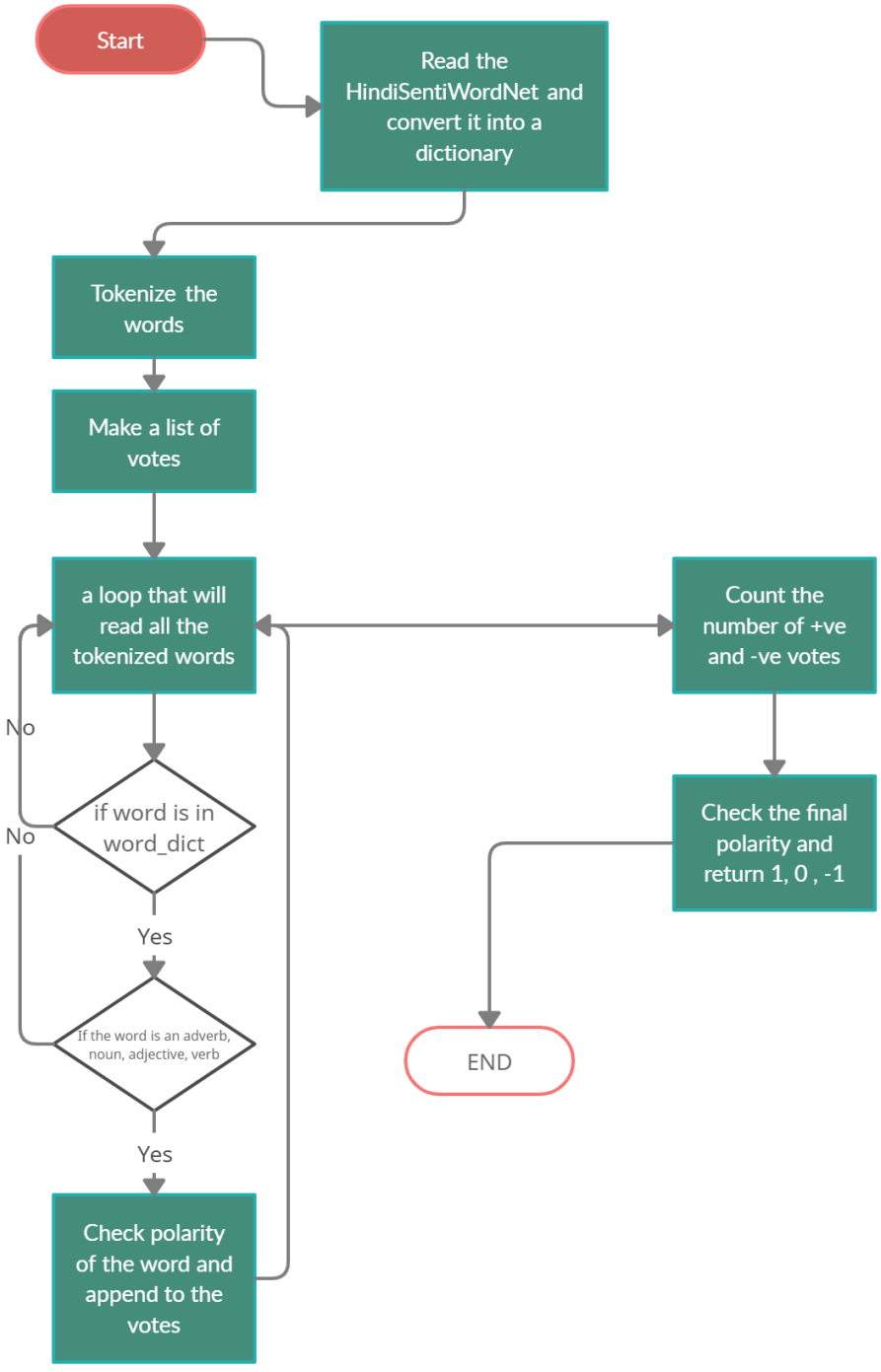
- Hindi SentiWordNet
- A dataset of positive and negative sentences that help us test our model.

Data-set used

We took the 250 sentences (125 positives and 125 negatives) based on movie reviews from the dataset provided to us by the TAs and more sentences from random sources on the internet, a link to which is here. From this dataset, we tried to select the few neutral sentences that were in these 2 datasets, and in total, we have more than 900 sentences in our dataset. The dataset can be found here.

ALGORITHM:

The flowchart given below shows the process we adopted in this program for sentiment analysis.



Program Description and Working:

- Read the HindiSentiWordnet.txt as a csv using pandas and read the pos_tags, ID of words, Positive/Negative score, and the list of words that correspond to that particular field.
- Make a dictionary of all the words in the above text file and store them in words_dict along with their pos tags and +ve and -ve scores. This is done because in many cases there are more than 1 words corresponding to a pos tag and +ve/-ve score, so to make calculations easier, we are making this word_dict, which is a dictionary of words which consist of each and every word in our sentiWordnet.
- Make a **Senti()** function, that is used to calculate the sentiment value of the text that is provided to it as an argument. We make a list named **votes** that is used to calculate the number of +ve, -ve, and neutral sentences in our dataset. We only allow words that are present in our sentiWordnet (that is, words with sentiment values) to enter the loop, and then we check the pos and neg, that is, the positive and the negative score of the particular word and if pos > neg, we increase the positive polarity (which is 0 at the start of the program) with pos and append 1 to the **votes** list to show that this is a positive word. Similarly, we do the same for negative words. Then we count the number of 1s and -1s in the **votes** list and if positive votes > negative votes then we return 1 and if negative votes > positive votes then we return -1
- Make a **condition()** function that takes 2 arguments; text and seen. 'seen' is taken from the above function. In this function, we check if the sentence text has the negating "नहीं", the inverting conjunction "लेकिन", and the preposition "पर". This function uses many subcases in order to filter-out the appropriate outcome. Furthermore, in case more than 1 of these words (नहीं,लेकिन,पर) occur, the highest precedence has been given to the negation, followed by the conjunction
- Make a code snippet that calculates the accuracy score and the F-measure for the entire sample data. For this, we use the sklearn library of python and which helps us calculate both of the above-mentioned data. We are splitting the sentences based on the presence of "#".
- Call the functions **Senti()** and **condition()** from main and run the program for the sample sentences (and the data)

ANALYSIS:

- The lack of words in the Hindi SentiWordnet is one of the main causes of the decrease in the accuracy of the overall program. The greater semantic diversity of all the words, in general in the Hindi language, gives rise to another problem. Consider the example of the word “पर” having three distinct meanings -
    - A preposition (*on*)
    - Synonym “to” (*but*)
    - Feather (*of a bird*)

  This has been a challenge to sort out the word in its distinct senses, and for a great proportion of the text.

- Hindi as a language, having a relatively free word order (not the strict S-V-O as in English) makes it difficult to normalise or create rules that satisfy a majority of the text. So accordingly, changes have been made to the code, the dataset as well as the Hindi SentiWordNet.
- Earlier the approach was to try to stem the Hindi words, which would have greatly reduced the number of words to be tallied with the Hindi SentiWordNet, had it worked. We faced many challenges at this point - Stemmer for Hindi had low accuracy, many times the word root had a different sentiment from the word (sometimes even inverted). Due to these reasons, we decided to drop the idea, which consequently gave us higher accuracy.
- Earlier, while tagging only the positive and negative sentiments (not considering neutral) in the data set, the accuracy score without making any changes to the Hindi SentiWordNet was 52.34%, which later on while considering all positive, negative, and neutral sentiments reduced to 36.13% as the sentences which were getting tagged by the program (predictive tagging as +1,0 and -1) but the actual tagging of the sentences was done only in +1 and -1 due to the unavailability of the neutral dataset at that point. After filtering neutral sentences from the dataset and modifying the SentiWordNet, the following are the milestones we reached in the calculation of the accuracy score:

```
200 sentences -->  42.64
250 sentences -->  43.34
350 sentences -->  45.33
475 sentences -->  45.93
```

This analysis is only for the negative sentences and on adding positive words to the SentiWordNet, we observed yet another surge in the increase in accuracy :

```
all sentences without the "नहीं" condition -->  59.177

all sentences without the "नहीं" condition and without stemming
-->  59.277
```

- The program before adding the special rules (conjunction, negation and preposition) gave a higher accuracy of 59.27% but on adding these rules, the accuracy dropped to 56.46%. We speculate this result because of the skewed dataset.
- For many sentences, the absence of contextual analysis proved to be a great hurdle as our sentiment analysis is totally lexicon-based, with no preface to semantics. We tried to bridge this gap by adding special rules that **could** help us also take into consideration the context of the sentence by processing the text into phrases, that are sliced at the position of a particular word (eg. नहीं,लेकिन,पर) and check the sentiment of each sliced phrases and get the score accordingly (according to lexical sentimental analysis)

CHALLENGES FACED:

- Hindi Senti wordnet is still in the developing phase and it still does not contain many words that denote sentiment in the program.
- In the case of rule-based modal, we could not consider the context of the sentences before giving them a Sentiment score because of the unavailability of a proper parser that would help us make Phrase structure trees in the Hindi language.
- Many wordforms were not available in the Hindi Senti Wordnet for example word "अच्छा" was present but the words "अच्छी" and "अच्छे" were not present in the wordnet.
- Idioms could not be considered in this project because the score is given based on senti wordnet and senti wordnet only considers words.
- The presence of homonyms in the senti wordnet caused ambiguity, for example in the sentence "वह कल हार गया" here "हार" which means 'to lose' and thereby the sentence should be rated as "negative" but in the senti wordnet, the word exists as a noun which means "A Necklace" and thereby rating the sentence as 'neutral'. If there was a function, that could consider the context of the sentence and check if the word in question is a noun, adjective, adverb, or verb, we could easily increase the accuracy of the given model as would only consider the word with that pos tag and not check "noun" in case of "हार" when it means a verb.

REFERENCES:

- https://sci-hub.do/https://ieeexplore.ieee.org/document/7379415
- Resource Centre for Indian Language Technology Solutions(CFILT) (iitb.ac.in)
- Sentiment Analysis of Transliterated Texts in Hindi and Marathi Languages | by Mohammed Arshad Ansari | Towards Data Science
- yash8589/CL1-Project-2: Sentiment Analysis (github.com)