# Sentiment Analysis And Transfer Learning

NNTI Project

Priya Priya, Yamini Supriya
7008117, 7010131
prpr00001@stud.uni-saarland.de , yave00002@stud.uni-saarland.de

**Abstract**

Sentiment analysis refers to the task of natural language processing to determine whether a piece of text contains some subjective information and what subjective information it expresses, i.e., whether the attitude behind this text is positive, negative, or neutral. Understanding the opinions behind user-generated content automatically is of great help for commercial and political use, among others. The task can be conducted on different levels, classifying the polarity of words, sentences, or entire documents. Whereas, Transfer learning is a machine learning technique where a model trained on one task is re-purposed on a second related task. It is an optimization that allows rapid progress or improved performance when modeling the second task. In this project, we implemented a neural sentiment classifier which classifies sentiments and detects hate speech in Hindi and Bengali dataset.

Github Link : https://github.com/priya00priya/NNTI-WS2021-NLP-Project.git/

## 1 Introduction

Nowadays, NLP is booming and used in different fields. It deals with the interaction between computers and humans using natural language. NLP is collaborating with different techniques like transfer learning, attention mechanism, etc. Syntactic analysis and semantic analysis are the main techniques used to complete Natural Language Processing tasks.

Our project is to create a neural sentiment classifier by using the HINDI HASOC dataset. We are provided with a Hindi dataset  a Bengali dataset. And we start training the word embeddings of the Hindi dataset, which helps in tracking the emotions presented in the Hindi dataset. And then we used this classifier on the Bengali dataset by using a technique called "Transfer Learning" where we consider the learned knowledge from the Hindi dataset. Finally, we tried to improve the accuracy of the model either by changing the model architecture or by changing the data representation.

# 2 Hindi Hasoc dataset task description and data construction

## 2.1 Getting the data

At first, we loaded the Hindi dataset which we downloaded from github(hindidataset.tsv). Then, we download the predefined stopwords for the Hindi language(stopwords.txt)

## 2.2 Data preparation

In this, we preprocessed the data by normalizing the text into lower case(because we want to limit ourselves just to 52 Hindi alphabets, and if we don't use this function then we have to deal with 104 alphabets which will increase the complexity in the model). And also we removed the punctuation symbols, stopwords, and words starting with '@'.

Hashtags and emojis : We kept the Hashtags(but removed the '' symbol). It is useful for sentiment analysis. Negative hashtags have an impact on the sentiment of the text. Emojis are removed because they are not important(according to us) and difficult to work with while building the model

Empty Values : After the above filtration, some rows of data were empty or spaces. Here we removed those unnecessary rows.

## 2.3 Build the vocabulary

Building the vocabulary helps in pre-processing of corpus text. We build the vocabulary by creating a list that contains all the unique words. In the same function, we calculated the frequency of the words as well. We stored it in the dictionary where the word is the key and frequency is the value.

## 2.4 Subsampling

Subsampling is also one of the techniques that we use when we are building word pairs, as we know these word pairs are sample training data.

Words with less frequency or infrequent words appearing as context words could be discarded as they may not provide contextual information to the central word.

The probability of keeping the word is defined as follows:

$$P_{keep}(w_i) = (\sqrt{z(w_i)/.000001} + 1) * (.000001/z(w_i)) \tag{1}$$

Where $z(w_i)$ is the relative frequency of the word.

$P_{keep}(w_i)$ is the probability of each word to be in the context

We did the subsampling on the data where we defined a function that takes the integer value of the word frequency as an input and returns the probability to keep the word in the context.

### 2.5 Skipgrams

Skip-gram is used to predict the context word for a given target word. Here, the target word is input while context words are output. It works well with a small amount of the training data, and represents well even rare words or phrases.

### 2.6 Hyperparameters

Hyperparameter is a parameter whose value is used to control the learning process. It is important because they have a significant impact on the performance of the model being trained. Initially we have chosen window size = 2 ,embedding size = 100, learning rate = 0.05, epochs = 100.

In addition to that, we initialized two weight matrices of word2vec. Softmax converts logits into probabilities. And also we choose "Binary Cross Entropy" and Adam(Adaptive Moment Estimation Algorithm) as loss function(criterion) and optimizer. Because Cross entropy(BCE) typically plays well with probability distribution outputs. It finds the distance between the truth value and the predicted value. And Adam is good with sparse data and deals with noisy data very well.

## 3 Hindi dataset Binary classifier

Long short-term memory networks (LSTMs) have gained good performance in sentiment analysis tasks. The general method is to use LSTMs to combine word embeddings for text representation. Only using word embeddings to represent words is inaccurate in sentiment analysis tasks. To solve the problem, we propose a lexicon-enhanced LSTM model. The model first uses sentiment lexicon as an extra information pre-training a word sentiment classifier and then gets the sentiment embeddings of words including the words not in the lexicon. Combining the sentiment embedding and its word embedding can make word representation more accurate.

Furthermore, we define a new method to find the attention vector in general sentiment analysis without a target that can improve the LSTM ability in capturing global sentiment information.

In this work, we present a new RNN model based on the self-attention mechanism to improve the performance while dealing with long sentences.

| Class | No. of Sentences | Percentage |
|---|---|---|
| **HOF** | 2469 | 53 |
| **NOT** | 2196 | 47 |

**Table 1:** HOF and not HOF - Task1

# 4 Bengali dataset task description and data construction

In preprocessing the Bengali dataset, we used the same approach as explained above. We have reproduced the Subtask A from the Hasoc paper. Here we started splitting the Bengali dataset and sampled randomly. We splitted the dataset in such a way that the part of Bengali dataset is equivalent to the Hindi dataset. Then, applied the preprocessing pipeline from above word embeddings tasks to the new data.

# 5 Results

| Model | Accuracy | Precision | Recall |
|---|---|---|---|
| LSTM- HINDI | 00 | 00 | 00 |
| LSTM- BENGALI | 00 | 00 | 00 |

**Table 2:** Results

# 6 Methodologies to improve the existing results

To enhance the performance and accuracy of a text classification model we have the following methods:

1. Domain Specific Features in the Corpus:- For a classification problem, it is important to choose the test and training corpus very carefully. For a variety of features to act in the classification algorithm, domain knowledge plays an integral part.

2. Use An Exhaustive Stopword List:- Stopwords are defined as the most commonly used words in a corpus. Most commonly used stopwords are "a, the, of, on, ... etc". These words are used to define the structure of a sentence. But, are of no use in defining the context. Treating these types of words as feature words would result in poor performance in text classification. There are some other supporting words as well which are of lesser importance than any other terms. These includes: Language Stop Words – a, of, on, the ... etc Location Stopwords – Country names, Cities names etc Time Stopwords – Name of the months and days (january, february, monday, tuesday, today, tomorrow ... ) etc Numerals Stopwords – Words describing numerical terms ( hundred, thousand, ... etc)

3. Noise Free Corpus:- In most of the data science problems, it is recommended to undertake a classification algorithm on a cleaned corpus rather than a noisy corpus. Noisy corpus refers to unimportant entities of the text such as punctuations marks, numerical values, links and urls etc. Removal of these entities from the text would increase the accuracy, because the size of sample space of possible features set decreases.

4. Normalized Corpus:- Words are the integral part of any classification technique. However, these words are often used with different variations in the text depending on their grammar.It is always a good practice to normalize the terms to their root forms. This technique is known as Lemmatization.

5. Use Complex Features: In some cases, features as the combination of words provides better significance rather than considering single words as features. Combinations of N words together are called N-grams. It is known that Bigrams are the most informative N-Gram combinations. Adding bigrams to the feature set will improve the accuracy of the text classification model.

6. Treat missing and Outlier values: The unwanted presence of missing and outlier values in the training data often reduces the accuracy of a model or leads to a biased model. It leads to inaccurate predictions. This is because we don't analyse the behavior and relationship with other variables correctly. So, it is important to treat missing and outlier values well.

7. We can improve the accuracy by choosing the appropriate model architecture. CBOW/skip gram for large corpus, higher dimensions for small corpus, faster use CBOW

8. By increasing the training dataset, vector dimensions, window size also helps in improving the accuracy.

# 7 References

https://www.oreilly.com/library/view/python-natural-language/9781787121423/f7035ac3-7624-4b80-b464-64ed8a7f252a.xhtml

https://lct-master.org

Chapter 11: Transfer Learning, Handbook of Research on Machine Learning Applications, 2009.

https://analyticsindiamag.com/how-to-create-a-vocabulary-builder-for-nlp-tasks/

https://www.youtube.com/watch?v=UqRCEmrv1gQ

https://nathanrooy.github.io/posts/2018-03-22/word2vec-from-scratch-with-python-and-numpy/

https://www.analyticsvidhya.com/blog/2015/10/6-practices-enhance-performance-text-classification-model/

https://ieeexplore.ieee.org/abstract/document/8513826