Hate Speech Detection (Sentiment Analysis)

A Project Report

Submitted for Minor Project (6CS191) of 6th Semester for partial fulfillment of the requirements for the award of the degree of

Bachelors of Technology

in

Computer science and Engineering

Submitted by

Sujit Chaurasiya (1706071)
Kartik Mohan (1706072)
Jitendra Kumar (1706073)
Sajal Agrawal (1706129)

Under the Supervision of

Prof. Kumar Abhishek

Asst. Prof. CSE Department
NIT Patna

Prof. Jyoti Prakash Singh

Asst Prof & Head CSE Department
NIT Patna

Department of Computer Science & Engineering

National Institute of Technology Patna-800005

Jan-June, 2020

राष्ट्रीय प्रौद्योगिकी संस्थान पटना
NATIONAL INSTITUTE OF TECHNOLOGY PATNA

CERTIFICATE

This is to certify that **Sujit Chaursiya Roll No. 1706071, Kartik Mohan Roll No. 1706072, Jitendra Kumar Roll No. 1706073, Sajal Agrawal Roll No. 1706129** has carried out the Minor project (6CS191) entitled as "Hate Speech Detection (Sentiment Analysis)" during their 6th semester under the supervision of Dr. Kumar Abhishek and Dr. JP Singh, of CSE Department in fulfillment of the requirements for the award of Bachelor of Technology (B.Tech) degree in the department of Computer Science & Engineering, National Institute of Technology Patna.

……. ……..……

Prof. Kumar Abhishesk
Assistant Professor
CSE Department
NIT Patna

Prof. JP Singh
Head of Department
CSE Department
NIT Patna

राष्ट्रीय प्रौद्योगिकी संस्थान पटना
NATIONAL INSTITUTE OF TECHNOLOGY PATNA

DECLARATION

We students of 6th semester hereby declare that this project entitled "**Hate Speech Detection (Sentiment Analysis)**" has been implemented by our group in the Department of Computer Science & Engineering of NIT Patna under the guidance of **Prof. Kumar Abhishek** and **Prof. JP Singh**, Department of Computer Science and Engineering, NIT Patna. No part of this project is submitted for the award of degree to any other Institute.

Name	**Signature**
Sujit Chaursiya	.....
Kartik Mohan	.....
Jitendra Kumar	.....
Sajal Agrawal	.....

Place	**Date**
NIT PATNA	.....

राष्ट्रीय प्रौद्योगिकी संस्थान पटना
NATIONAL INSTITUTE OF TECHNOLOGY PATNA

ACKNOWLEDGEMENT

We would like to acknowledge and express my deepest and affectionate gratitude to our mentors **Prof. Kumar Abhishek** and **Prof. JP Singh**, of Computer Science & Engineering Department, National Institute of Technology Patna for the valuable guidance, sympathy and co-operation for providing necessary facilities and sources during the entirety of this project.

We wish to convey our sincere gratitude to the Head of Department and all faculties of the Computer Science & Engineering Department who have enlightened and motivated us during our studies. The faculties and cooperation received from the technical staff of the Department of Computer Science & Engineering is sincerely acknowledged.

1. Sujit Chaurasiya (Roll No. 1706071)
2. Kartik Mohan (Roll No. 1706072)
3. Jitendra Kumar (Roll No. 1706073)
4. Sajal Agrawal (Roll No. 1706129)

Contents

Page No.

I. Certificate 2
II. Declaration 3
III. Acknowledgement 4
IV. Contents 5

1. **Abstract** 6
2. **About the Project**
   - 2.1 Introduction 7
   - 2.2 Task Definition 7
3. **Preprocessing** 10
4. **Embeddings** 11
5. **Model Classification and Evaluation**
   - 5.1 CNN 13
   - 5.2 LSTM 13
   - 5.3 BERT 13
6. **Result** 15
7. **Discussion** 17
8. **Conclusion** 19

1. Abstract:

Hate Speech Detection is such a topic that has been gaining quite a traction in the past few years due to its uniqueness and challenging to identify property and also it impacts a society as whole in a very different way. Any attack direct or indirect towards a group of people or race or gender or religion of people can be coined as Hate Speech. When it takes the form of verbal and textual communication then we regard it as online bullying which is also a cyber crime. But due to various loose and unclear laws it usually is not identified and thus remains a big problem in today's world. We had an objective of segregating the data as hate speech or not hate speech. The various datasets obtained from Stormfront (a white supremacist forum) for English language. Then we had to study the given data and pre-process it accordingly so that it runs on specified machine learning models.

Also the main focus was to improve the hate speech detection capacity of our project so this includes a various number of models with appropriate embeddings which helped in finding the best possible model and also lays the groundwork for all the things which can be used and implemented in the future to overcome the faults of these models and give a more precise result. We were expected to deliver a working model with an accuracy metric high enough which can separate or classify the data into the respective classes.

2. About the Project:

2.1 **Introduction**:

Many of the social networking platforms of today are suffering from the problem of online harassment and bullying which is coined as Hate Speech. Increasing the number of users makes it a very hard and unapproachable job to determine it, also due to the vagueness of these terms from person to person, it becomes a tedious challenge in itself to identify and segregate it as a complete and non debatable topic.

Hate speech or comment is determined when any message targets a community or group of people and uses negative or inappropriate words or comments regarding them, which leads to cyberbullying and ill use of the internet to spread chaos. But now in the current age with the help of various techniques and learning models, we have tried to classify and segregate these offensive messages and label them as hateful or not. The machine learning models were applied to determine comments of three languages, English, Hindi and Bangla (in many instances the languages were not pure as certain multilingual words and phrases also made it to the dataset thus complicating the process of determination, which was a challenge in itself).
The focus was on determining the tweets/comments by their actual labels and possibly one of the biggest challenges that we encountered was the incorrect labelling of data since the naïve models were not able to gather context of the sentences and labelled any data as Hate if it contained a negative word. So, it was very important to overcome this situation and to recall it more accurately.

2.2 **Task Definition**

Our techniques and ideas were heavily inspired by the paper ‘Hate Speech Dataset from a White Supremacy Forum’ by Ona de Gibert, Naiara Perez, Aitor Garc´ıa-Pablos, Montse Cuadros. A major portion of our dataset of English was obtained from their dataset which was collected by web scraping the Stormfront website.

Collection of dataset and labelling it as Hate Speech or not was one of the bigger obstacles as it is without a complete and stable definition. Many people have their own perception of hate speech as a whole and do not abide by the definitions put forth by the other entity which results in misjudgement. Identifying any text or document as hateful is a big task to accomplish.
Our dataset consisted of two subtasks, Subtask A which classifies the data into one of the three labels NAG (Non_aggressive), CAG (Covertly_aggressive) and OAG (Overly_aggressive), so it goes another level deeper in classification that is it not only classifies the data as hate speech it also tells about the extent to which the data is offensive and Subtask B which classifies the data into one of two labels 'Hate' and 'NoHate', which is simple binary classification of the dataset.
Alternatively, Subtask B had another label as 'Skip', for the cases which cannot be determined as hate or no hate. But in our model, they were not useful and hence were omitted from our training dataset.
After combining all the different datasets from all the web forums and platforms we effectively had the following datasets:

- English training dataset (21436 training samples for Subtask A and 4263 training samples for Subtask B)
- English validation dataset (1066 testing samples)
- Hindi training dataset (10956 training samples)
- Hindi validation dataset (998 testing samples)
- Bangla training dataset (3827 training samples)
- Bangla validation dataset (927 testing samples)

English dataset had limited training samples for Subtask B, since the effective dataset file was combined from different files eng_train.csv and four separate csv file (agr_en_dev.csv, agr_en_fb_gold.csv, agr_en_train.csv, agr_en_tw_gold.csv) scrapped from different websites.

Hindi dataset files hin_train.csv, hin_test.csv and three separate csv file (agr_hi_dev.csv, agr_hi_fb_gold.csv, agr_hi_train.csv) were combined to form the dataset. And Bangla dataset consists of iben_dev and iben_train.

The task was to find and train the most suitable machine learning models on these given training files and test them on the given validation sets. However few things to be kept in mind are that the data is very noisy that is there are a lot of samples that are either meaningless or have words from different languages which makes it a very challenging task to exactly determine the context of the given comments.

Also, the multilingual element of the dataset makes it harder to form the vocabulary of it as there are many words present which are unknown to a particular language and thus need some pre-processing or other embedding techniques so as to correctly determine the labels of the dataset.

3. Preprocessing

The datasets were combined and were ready to be used to train the model but before that several pre-processing and alterations needed to be done to the data so that it can be easily understood by the machine and as well as make sense out of it. This included converting the data from letters to numbers and filtering the obvious candidates which contributed nothing to the task definition and were just present as noise in our dataset.

The following algorithms which were involved in data transformation and data filtering are:

- **Text cleaning**- Using python regular expression and string library, we converted the whole dataset in lower alphabet case. We removed words symbols like “?,.,’ ”” etc. Then we removed punctuation symbols and words having digits in them.
- **Removal of stop words**- Using neural language tool kit we tokenized text sentences and removed stop words from the text sentences.
- **Tokenization**- Then we made a tokenizer based on most common 5000 words and fit it on text sentences. **Tokenized** which implies that all of the sentences were broken down into individual words. Basically, a dictionary was created where each word was assigned a specific value and the length of this *word_index* (Dictionary of sorts) was the size of our vocabulary, which is represented by *vocab_size* in our model. But the emoticons were not removed because when embeddings were used, there were individual vectors present for each emoticon thus providing more context to our data.
- **Post Padding**- After the necessary cleaning done, the average length of the dataset was calculated, which is the average length of a text in the entire file and it was found to be **30** for English dataset, **48** for Hindi dataset and **26** (which was rounded to 30) for Bangla dataset. Then **post padding** was done, which refers to the addition of 0’s to the end of an encoded text, to make the length of all the encoded texts the same. Therefore, we had our encoded train, dev and test files with the Tokenizer and were padded so all had the same length.

4. Embeddings

Apart from the basic filtering and transformation of data, embeddings were used as well. Embeddings are a set of pretrained vectors wherein each vector corresponds to a word present in the vocabulary of the language. Embeddings are generally preferred when the dataset is small or noisy and thus the already trained weights help in improving the accuracy as well as speed of the dataset prediction. There are plenty of embeddings used in machine learning but the ones we used were:

- **Fast-text:** Facebook's Artificial Intelligence Research team produced this library to let a machine learning model learn the word embeddings and do text classifications. This library helps to design a machine learning algorithm to acquire vector representations of tokens. The fast-text embedding was taken from https://fasttext.cc/docs/en/pretrained-vectors.html.

  There are fast-text embeddings present for most of the languages, however there is no embedding present for mixed languages so that is a problem which needs to be tackled in a different manner.
  Even though it requires more time to train a model with fast-text embedding because the number of n-grams are much greater than the number of the tokens, it gives preferable outcomes as compared with Word2Vec. Fast-text can represent infrequent words properly.

- **One-hot**: One-hot is a bunch of bits in which the valid combination only has a set bit and all other bits are unset. Now this embedding is created uniquely for each dataset as every dataset has a unique bag of words or vocabulary, so mixed language inputs are very well trained with this embedding but a drawback faced while using one-hot encoding is that the context or the meaning associated with the word is lost and it is treated as either being present or not in the sentence. Hence, the word loses its context and is held positive (one) if present or negative (zero) if not, that means all the words in the sentence are ones and all absences are zeros.

- **GloVe (Global Vectors for Word Representation):**

GloVe is an unsupervised learning algorithm developed at Stanford to create a word embedding by collecting global word-word co-event metrics from a corpus. The resulting embedding vectors show interesting linear substructures of the word in space. Word2Vec accepts texts as input to train a neural network. The resulting embedding checks do the words stand for similar contexts. GloVe focuses on co-events words throughout the corpus. Its embedding possibilities relate to that two words appear together. This was one of the primary reasons why Word2vec was not included as embedding.

5. Model Classification and Evaluation

Methodology:

The next step, after getting the embedding weight matrix and data transformation, is to fit that training data to a specific model and train it on all of the samples in the training dataset. There were several combinations of embeddings and models used with our datasets and each of the combinations yielded a different result, thus signifying that data is trained differently by each of the training models. The models which were used were:

- **Convolutional Neural Network (CNN):** An embedding layer was added in the Sequential model, with input dimension as *vocab_size*, output dimension as 8, input length as 30 and weights (if needed) as embedding matrix which contains the weights pre trained in the Fast-text file. Then we used a 1-dimensional convolution with 128 filters, kernel size 3 and activation function "ReLU". Then used a dropout layer with rate 0.5, followed by a Maxpool layer with 5 filters, followed by dense layer 1 with output 256 and activation function "ReLU", and dense layer with output 2 or 3 depending upon the subtask with activation function "softmax". Model was compiled with categorical/ binary_crossentropy with ADAM optimizer.

- **Long Short-Term Memory (LSTM)**: An embedding layer was added in the Sequential model, with input dimension as *vocab_size*, output dimension as 100, input length as 30 and weights (if needed) as embedding matrix which contains the weights pre trained in the Fast-text file. Then we used a LSTM Layer with 192 filters and recurrent dropout rate of 0.2, followed by a dense layer with output 2 or 3 depending upon the subtask with activation function "softmax". Model was compiled with categorical/ binary cross entropy with ADAM optimizer.

- **BERT**: Sentences are tokenized using FullTokenizer of BERT. Tokens are converted into sequences and padded up to 256 sequence length. Sequences are passed as input in the model. Then we used a BertModelLayer and loaded pretrained weights from file "uncased_L-12_H-768_A-12" which contains vocab and pretrained model trained by google research. After this layer we used a dense layer of 768 units.

For output we used a dense layer using softmax as activation function. Model is compiled using Adam optimizer and categotical_cross entropy as loss function.

Now the different embedding and model combinations which were used for different languages were. For English language, Hindi language and Bangla language:

i. Fast-text embedding with CNN for Subtask A and Subtask B
ii. One hot encoding with CNN for Subtask A and Subtask B
iii. Fast-text embedding with LSTM for Subtask A and Subtask B

Therefore, for each language there were six different models and embedding combinations (three for Subtask A and three for Subtask B).

6. Results

The results what we get will given in terms of the accuracy for all classes separately, as we have ‘hate’ and ‘noHate’ classes so we give results for these classes independently. For overall accuracy we define macro average which is calculated as the average of all the classes, which implies that all the classes are treated as equals.

Macro-average precision=(P1+P2)/2

Macro-average recall=(R1+R2)/2.

Bangla dataset:

Model Name and Subtask	Macro average
Fasttext-CNN-SubtaskA	0.58
Fasttext-CNN-SubtaskB	0.66
Onehot-CNN-SubtaskA	0.58
Onehot-CNN-SubtaskB	0.70
Fasttext-LSTM-SubtaskA	0.60
Fasttext-LSTM-SubtaskB	0.76
BERT Model SubtaskA	0.66
BERT Model SubtaskB	0.78

Hindi dataset:

Model Name and Subtask	Macro average
Fasttext-CNN-SubtaskA	0.55
Fasttext-CNN-SubtaskB	0.66

Onehot-CNN-SubtaskA	0.57
Onehot-CNN-SubtaskB	0.60
Fasttext-LSTM-SubtaskA	0.55
Fasttext-LSTM-SubtaskB	0.67
BERT Model SubtaskA	0.61
BERT Model SubtaskB	0.74

English dataset:

Model Name and Subtask	**Macro average**
Fasttext-CNN-SubtaskA	0.53
Fasttext-CNN-SubtaskB	0.62
Onehot-CNN-SubtaskA	0.52
Onehot-CNN-SubtaskB	0.64
Fasttext-LSTM-SubtaskA	0.50
Fasttext-LSTM-SubtaskB	0.62
BERT Model SubtaskA	0.57
BERT Model SubtaskB	0.71

7. Discussion

Here we will discuss why the specific models were chosen for training and what is the motivation behind them. Also, we will discuss why they perform so well with the dataset.

CNNs which were generally applied on image processing and other such vision tasks are also many times applied on text searchings and patterns, this is because it gives very good and hopeful output for Natural Language Processing (NLP) tasks. The output of each convolution fires if a unique arrangement of words is detected. By increasing or decreasing the sizes of the kernels and combining their outputs together, allows to detect patterns of multiple lengths. Patterns could be expressions for example “I think”, “Thank God”, “I don’t know” etc. and these can be identified in a CNN without knowing their positions in the sentence. However, this was not enough to determine the context of the sentence as it was not able to figure out the meaning of the given anagrams without looking at the phrases ahead towards the end of the sentence.

That’s when we thought of applying the LSTM model on our dataset. Long Short Term Memory, it is an addition upon our general RNNs because they are also extremely good at long term rememberings. They are designed in such a way that they are able to avoid such dependency with ease, and as a result of this they are able to retain information for a very long duration of time which helps us in pattern identification. However LSTMs are prone to overfitting and to deal with that problem dropout layers were added into the model, LSTM Models gave better results than CNN in almost all of the cases which supports the fact that our assumption of CNN models not getting the context of the statement was true and there were many data which needed correct context to be classified into their respective labels based upon the given subtasks.

But still one issue which was hindering in a perfect model was that the languages in either of the datasets was not pure and had other words,

phrases of other languages which hampered the models from understanding and predicting. So, we had to shift to a model which had multilingual weight training and can be used here as it would have been previously trained on huge corpus of data. The BERT model is the one which we implemented next.

BERT (bidirectional encoder representations from Transformers) is a deep learning model developed at Google. BERT Model has achieved state-of-the-art outcomes in 11 individual NLP functions. BERT is a bidirectional model which is based on Transformers architecture by replacing the neural networks having sequential nature (for example - RNN (LSTM and GRU)) with a speedy attention-based approach. The model was also trained on two unsupervised datasets, masked language modeling and next sentence prediction. A pre-trained BERT model can be used for text classification tasks by fine tuning it.

8. Conclusion:

The result files or datasets consist of sentences in the testing or validation dataset which are labelled as the hate speech and no-hate speech and also the level of the hateness for these sentences which are in the 3 class problem.

Among all the models which we are using in our project (CNN, LSTM, BERT Models) the most accurate one was the BERT model,which is closely followed by fast-text with LSTM.

Now,the reason for BERT as most accurate was that it was able to deal with multilingual words appearing in the sentences as it had letter from other languages available as well and also it works on efficient working of transformers and attention model and fast-text LSTM was a close competition because fast-text uses substring of anagrams for pattern searching which is much faster in getting context and meaning of any sentences of and LSTM is heavily successful in text classification.