Minor Project Report

Hate Speech Detection (Sentiment Analysis)

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

Submitted to- Dr. J.P. Singh and Dr. Kumar Abhishek

Department of Computer Science and Engineering

National Institute of Technology, Patna

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 partial fulfillment of the requirements for the award of Bachelor of Technology degree in the department of Computer Science & Engineering, National Institute of Technology Patna.

……..	…….
Dr. Kumar Abhishesk	Dr. JP Singh
Assistant Professor	Head of Department
CSE Department	CSE Department
NIT Patna	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 carried out by us in the Department of Computer Science and Engineering of National Institute of Technology Patna under the guidance of Dr. Kumar Abhishek and Dr. JP Singh, Department of Computer Science and Engineering, NIT Patna. No part of this project has been submitted for the award of degree or diploma to any other Institute.

राष्ट्रीय प्रौद्योगिकी संस्थान पटना
NATIONAL INSTITUTE OF TECHNOLOGY PATNA

ACKNOWLEDGEMENT

We would like to acknowledge and express my deepest gratitude to our mentors Dr. Kumar Abhishek and Dr. 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 entire period of this project. We wish to convey our sincere gratitude to the Head of Department and all the faculties of Computer Science & Engineering Department who have enlightened us during our studies. The faculties and cooperation received from the technical staff of Department of Computer Science & Engineering is thankfully 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)

Abstract:

Hate speech is commonly defined as any communication that disparages a target group of people based on some characteristic such as race, colour, ethnicity, gender, sexual orientation, nationality, religion, or other characteristic. Due to the massive rise of user-generated web content on social media, the amount of hate speech is also steadily increasing. Over the past years, interest in online hate speech detection and, particularly, the automation of this task has continuously grown, along with the societal impact of the phenomenon. We had a 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 classifying the data into the respective classes.

Introduction:

The rapid growth of content in social networks such as Facebook, Twitter and blogs, makes it impossible to monitor what is being said. The increase of cyberbullying and cyberterrorism, and the use of hate on the Internet, make the identification of hate in the web an essential ingredient for anti-bullying policies of social media.

Hate speech or comment is determined when any message targets a community or group of people and use 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.

Problem Definition and Algorithm:

1. 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 scrapping the Stormfront website.

Collecting and annotating data for the training of classifiers to detect hate speech is challenging. Specifically, identifying and agreeing whether specific text is Hate speech is difficult, as per previously mentioned, there is no universal definition of hate speech.

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 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 consisting 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.

2. Algorithm Definition-

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 whole dataset in lower alphabet case. We removed words symbols like “?,.,’ ’” etc. Then we removed punctuation symbols and words having digit in it.

- 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. **Tokenised** 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 as same. Therefore, we had our encoded train, dev and test files with the Tokenizer and were padded so all had the same length.

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: It is a library for learning of word embeddings and text classification created by Facebook's AI Research lab. The model allows to create an unsupervised learning or supervised learning algorithm for obtaining vector representations for words. The fast-text embedding was downloaded 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. Although it takes longer time to train a **Fast-Text** model (number of n-grams > number of words), it performs **better than Word2Vec** and allows rare words to be represented appropriately.

- One-hot: In digital circuits and machine learning, one-hot is a group of bits among which the legal combinations of values are only those with a single high bit and all the others low. 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 absent are zeros.

- GloVe (Global Vectors for Word Representation): **GloVe** stands for global vectors for word representation. It is an unsupervised learning algorithm developed by Stanford for generating word **embeddings** by aggregating global word-word co-occurrence matrix from a corpus. The resulting **embeddings** show interesting linear substructures of the word in vector space. **Word2Vec** takes texts as training data for a neural network. The resulting embedding captures whether words appear in similar contexts. **GloVe** focuses on words co-occurrences over the whole corpus. Its embeddings relate to the probabilities that two words appear together. This was one of the primary reasons why Word2Vec was not included as an embedding.

Experimental Evaluation:

1. 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 pretrained 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 pretrained 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 crossentropy with ADAM optimizer.

- **BERT**: Sentences are tokenized using FullTokenizer of BERT. Tokens are converted into sequences and padded up to length of 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 language 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).

2. Results-

The baseline experiments include a majority class baseline showing the balance between the two classes in the test set. The results are given in terms of accuracy for HATE and NOHATE individually, and the overall accuracy. A macro-average will compute the metric independently for each class and then take the average (hence treating all classes equally).
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

3. Discussion-

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

CNNs are generally used in computer vision, however they've recently been applied to various NLP tasks and the results were promising. The result of each convolution will fire when a special pattern is detected. By varying the size of the kernels and concatenating their outputs, you're allowing yourself to detect patterns of multiples sizes (2, 3, or 5 adjacent words). Patterns could be expressions (word ngrams?) like "I hate", "very good" and therefore CNNs can

identify them in the sentence regardless of their position. 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 LSTM model on our dataset. Long Short-Term Memory networks – usually just called "LSTMs" – are a special kind of RNN, capable of learning long-term dependencies. LSTMs are explicitly designed to avoid the long-term dependency problem. Remembering information for long periods of time is practically their default behaviour, not something they struggle to learn. 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 trainings 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 by Google. It represented one of the major machine learning breakthroughs of the year, as it achieved state-of-the-art results across 11 different Natural Language Processing (NLP) tasks. BERT is a bidirectional model that is based on the transformer architecture, it replaces the sequential nature of RNN (LSTM & GRU) with a much faster Attention-based approach. The model is also pre-trained on two unsupervised tasks, masked language modelling and next sentence prediction. This allows us to use a pre-trained BERT model by fine-tuning the same on downstream specific tasks such as sentiment classification.

Conclusion:

The result files or datasets consist of sentences in the testing or validation dataset labelled as conveying hate speech or not and up to the severity they convey hate. Since, the definition of hate speech has many subtleties, this work includes a detailed explanation of the guidelines and manual annotation. Furthermore, several aspects of the resulting dataset have been studied, such as the necessity of additional context by the annotators to make a decision (which was fulfilled to an extent by

BERT Model), or the distribution of the vocabulary used in the examples labelled as hate speech. Among all the models used the most accurate one was BERT, closely followed by fast-text with LSTM. Now the reason was that BERT was able to deal with multilingual words appearing in the sentences as it had letter from other languages available as well, also, it works on efficient working 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 sentence and LSTM is heavily successful in text classification.