Hate Speech Detection in Roman Urdu on Twitter

By

Muhammad Ali Shahzad (21100185)

Saad Bin Waqas (21100241)

Iman Aleem (21100245)

Senior Project

2021

Department of Computer Science

Syed Baber Ali School of Science & Engineering Lahore

University of Management Sciences (LUMS)

CERTIFICATE

I certify that the senior project titled “Hate Speech Detection in Roman Urdu on Twitter” was completed under my supervision by the following students:

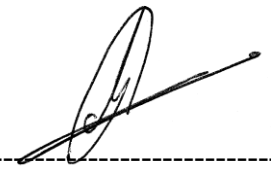
Muhammad Ali Shahzad (21100185)

Saad Bin Waqas (21100241)

Iman Aleem (21100245)

---

and the project deliverables meet the requirements of the program.



---------- Date: 19 May, 2021

Advisor (Signature)

---------- Date:

Co-advisor (if any)

Project Abstract

People all over the globe use social media as an outlet to exercise their freedom of speech and engage in discourses over literally every topic. These discussions are, however, not always positive and polite, and do end up taking negative directions. One of the most widely used platforms for expression of one's opinions is Twitter, where millions of users indulge in conversations on a daily basis, over not only day to day issues, but sensitive and controversial topics as well. The use of profane language and offensive words is referred to as hate speech. In recent years, a considerable amount of work has been done on the detection of hate speech on Twitter, using Natural Language Processing (NLP) techniques, not only on English tweets, but on tweets in a variety of different languages. In our project, we created a novel dataset, extracting Roman Urdu tweets from twitter, and ran multiple models to classify these tweets into 4 categories: normal (not hate speech), profane, sexist, religious hate. We took inspiration from another project, in not only choosing our labels, but also the embeddings, as well as tweets from the same project.

Acknowledgement and Dedication

The authors wish to express sincere appreciation to our close friends and family, especially Hafsa, Abdul Hadi, Rafay, and Wardah for their constant support during the course of this senior project and in the preparation of this manuscript especially in these difficult times. Special thanks to Dr.Agha, Moughees and Haris for their valuable input throughout this project. It would not have been possible without any of you.

Table of Contents

List of Figures

List of Tables

Chapter 1

1. Introduction

This project performs hate speech detection on Roman Urdu tweets, and classifies them into 4 distinct classes. The tweets it labels are collected from Twitter, and a dataset created in a similar project has also been made use of. People engage in all sorts of discourses and expressions online, and many times, the speech used is offensive and targets particular audiences. The detection of hate speech, particularly its classification and narrowing it down to particular classes is essential to ensure a safe and hate-free environment for online users.

1.1. Problem Statement

This project contributes towards detecting if particular tweets belong to hate speech or not. By collecting tweets from Twitter, we manually label them as belonging to 4 non-overlapping classes of hate speech. Once the dataset has been annotated, the project builds a model that classifies tweets into different categories of hate speech.

1.2. Social Benefits And Relevance

A safe and healthy online environment is a fundamental right of every internet user. Hate content has the potential not only to incite violence, but also have drastic effects on the mental health and wellbeing of those being targeted. Fights and conflicts over posts, threats, all these may trigger chains of violent events, and this may have disastrous effects on the society as a whole.

Therefore, in order to maintain peace and regulate offensive content on social media sites such as Twitter, it is essential to keep a check and more importantly, to be able to classify hate speech into distinct classes, so that relevant action can be taken. A model that has the ability to group tweets into different categories of hate speech could be of great use in regulating content on Twitter and keeping a check on what is being circulated, spread and shared.

There are a number of ethical and legal concerns and challenges around this project. Some of them are listed below:

Representativeness. While extracting tweets from Twitter using filters of words and phrases etc, we need to ensure that the samples generated as a result of these filters are more holistic and represent the realities of a diverse group of users, not only a particular group.

Privacy. The content collected may, in many cases, belong to users who do not want their identities to be revealed. Even if the usernames are discarded during the labeling process, the content they may not want to be exposed would have to be used.

Subjectivity. In many cases, there seems to be a lot of ambiguity around the definitions of hate speech and those of different classes. One group carrying out a project may classify a particular tweet as, for example, sexist, while another group may classify the same one as profane, or as any other class that does not even belong to the list of classes defined by the first group. This may introduce bias.

1.3. GOALS

The major goal of the project is to generate a dataset by extracting and self annotating Roman Urdu tweets. Furthermore we also aim to explore multiple machine learning algorithms that classify tweets from this dataset into our chosen classes/categories of hate speech, in order to find the optimum model with the best performance on our given roman urdu dataset.

1.4. OBJECTIVES

In order to achieve the primary goals of the project, the following objectives were defined and followed along its course:

1. Scrape raw tweets from Twitter in order to generate a large dataset.
    - Randomly select keywords/phrases from the hate speech lexicon/dictionary
    - Search for tweets using these randomly selected words as filters

- Data clean the tweets by removing usernames and links/URLs from the tweets

2. Label each tweet in the dataset as belonging to one of the 4 classes defined
    - Discard the tweets that are based on single words
    - Discard the tweets that are in any language other than Roman Urdu (English, Turkish etc.)
    - Tweets that raise ambiguity as to which class they belong to, are to be kept aside for later review and labeled separately by each annotator to reach a unanimous agreement on its label
3. Training our machine learning models
    - Implement different models with different preprocessing techniques
    - Train these models to predict the labels for the tweets
    - Aim for a high accuracy and weighted average f1 score

1.5. Deliverables (Outputs)

The output of this project is a manually labeled novel dataset of tweets in Roman Urdu, with the tweets classified into 4 distinct classes. Additionally, this report also serves as a deliverable, explaining the details of the different stages of the project in thorough detail.

Chapter 2

2. Background

2.1. Literature Review

Extensive work on the topic of hate speech detection has already been done, especially in the English language. Different methods have been attempted by various people in successfully classifying a text as hate speech or not. Watanabe *et al.* [1] make use of unigrams and patterns detected automatically while training their dataset, which are then used as features for training their machine learning model. Their algorithm was tested on a dataset of 2010 tweets and gave an accuracy of 87.4% for binary classification of whether text is offensive or not, and an accuracy of 78.4% for ternary classification of whether text is hateful, offensive, or clean. Gaydhani *et al.* [2] worked on a comparative analysis of different machine learning models with multiple varying parameters to see which model performs the best. They found that out of Logistic Regression, Naive Bayes, and Support Vector Machines, Logistic Regression performed the best given a parameter setting of n-gram range 1-3 with L2 regularization of TF-IDF. A Recurrent Neural Network approach was used by Pitsilis *et al.* [3] which incorporated user tendency towards racism and sexism as well as word frequency vectorisation for implementing features. They trained their data on a corpus of 16k manually labeled tweets and found that the RNN model performed better than the other state-of-the-art models by giving an F-score of 0.9320.

In [4], Waseem *et al.* annotate more than 16,000 tweets and find the best performing features by conducting a grid search over different feature combinations. Bootstrap sampling test was done to measure the statistical significance between feature sets. It was also found that a solid foundation could be achieved with a character n-gram based approach. [5] explores an under-discovered arena in hate speech detection - topic-dependent hate speech, by observing user behaviour and the spread of hate speech via retweets. The initiation of hate speech for any given hashtag is predicted and multiple models are suggested. Also, a neural architecture is proposed

that uses scaled dot-product attention and achieves a high macro F1-score, defeating multiple state-of-the-art models.

Alatawi *et al.*, in [6], explore the automatic detection of white supremacist hate speech on Twitter using two approaches. The first one extracts domain-specific embeddings from white supremacist corpus, to catch meanings of slangs using bidirectional Long Short-Term Memory (LSTM) deep learning model The second approach uses BERT. Tested on a balanced dataset, the first and second approaches reach F1-scores of 0.74890 and 0.79605 respectively.

Work has been done in other languages as well, such as German [7], Spanish [8], Indonesian [9], in [10], Pelle *et al.* use Hate2Vec based on classifier ensemble techniques of logistic regression and bag-of-words. For hate speech detection in Danish [11], Sigurbergsson and Derczynski worked with four different machine learning models including Logistic Regression, Learned-BiLSTM, Fast-BiLSTM, and AUX-Fast-BiLSTM. [12] used a dataset of YouTube comments in Arabic and trained a Support Vector Machine classifier. After experimenting with several preprocessing methods and different combinations of word-level and N-gram features, the classifier achieved an accuracy of 90.05% and it was revealed that the use of both stemming and N-gram features simultaneously in the same machine learning model is not beneficial.

The detection of hate speech in the German language by [7] used neural networks to train classifiers, with challenges arising from the short lengths of the tweets and the annotated categories being quite close. In Indonesian [9], Ibrohim *et al.* use Naive Bayes, Support Vector Machine, and Random Forest Decision Tree classifier with simple word n-gram and char n-gram features, to classify tweets into 3 classes - not abusive, abusive but not offensive, offensive. Experiments revealed that Naive Bayes performed better than the Support Vector Machine and the Random Forest Decision Tree, for the classification task, with word unigram and the combination of word n-gram giving better feature extraction results.

[8] introduces an intelligent system *HaterNet* that analyses hate speech evolution on Twitter using social network analysis techniques. A dataset consisting of 6000 labeled tweets in Spanish is also introduced. After analysing several techniques, a combination of a LTSM+MLP neural network, that obtains an area under the curve (AUC) of 0.828 on the dataset, is found to outperform previously existing methods.

At the same time, Roman Urdu has also been under the attention of work done on hate speech detection. In [13], Akhter *et al.* present a dataset containing user-generated comments online. Extracting features at word and character level, a total of 17 classifiers have been used to detect hate speech in the comments in Urdu as well as Roman Urdu. Achieving 99.2% and 95.9% values of F-measure on Roman Urdu and Urdu datasets respectively, LogitBoost and SimpleLogistic were found to outperform other models, with character-level tri-gram performing better than the other word and character n-grams.

A similar project [14] by Mehmood *et al.* gathered 11,000 reviews in Roman Urdu from 6 different sources. To enhance state-of-the-art algorithms, experiments were conducted on word-level features, character level features, and feature union, with the best results reducing the error rate by 12%. T-test and confidence interval on the achieved results proved the best results of each study to be statistically significant from the baseline. However, lack of pre-trained Roman Urdu word embeddings as well as the complexity arising from the variety of spellings in Roman Urdu, served as a hurdle in achieving the best possible results.

Perhaps the most relevant and holistic attempt at hate speech detection is the project [15] by Dr. Asim Karim, which also serves as a major source of inspiration for our project. In addition to providing a manually labeled dataset of around 10,000 tweets in Roman Urdu, the project also presents a lexicon of hateful words in the language. Tweets are classified into 5 distinct labels and the feasibility of transfer learning of 5 embedding models in Roman Urdu is looked into. Embeddings are trained on 4.7 million tweets, and it is revealed that transfer learning proves to be superior as opposed to building embeddings from scratch.

Chapter 3

3. DATASETS AND EVALUATION MEASURES

3.1. PRE-EXISTING DATASETS FOR INITIAL MODEL TESTING

We began our project by initially working on pre-existing datasets publicly available on Kaggle. Below are the details of the datasets used:

English Tweets: This pre-existing dataset is based on around 30,000 tweets in English. The tweets have been assigned the binary labels of negative and positive.

Roman Urdu Dataset: A compilation of tweets in Roman Urdu, collected from e-commerce websites reviews, comments on pages and twitter accounts. Based on more than 20,000 tweets manually labeled, each tweet has a tag attached to it - negative, positive or neutral.

Hate Speech Roman Urdu (HS-RU-20): This dataset consists of around 8,570 tweets, classified into 2 levels. The first level labels tweets as either neutral or hostile (5000 tweets), while the second level classifies the hostile tweets into offensive and hate speech (3570 tweets).

Hate Speech and Offensive Language Dataset: Dataset of 24783 tweets. The text is classified as: hate-speech (0), offensive language (1), and neither (2)

Table 1 summarises relevant information about the above explained datasets.

Table 3.1 DATASETS USED FOR INITIAL MODELS

Dataset	Size	Features	Labels	Source
English Tweets	~30 000	English Text	Negative, Positive	Kaggle
Roman Urdu DataSet	~20 000	Roman Urdu Text	Negative, Positive, Neutral	Kaggle
Hate Speech Roman Urdu (HS-RU-20)	8500	Roman Urdu Text	HateOffensive, NeutralHostile	Kaggle
Hate Speech and Offensive Language Dataset	~25 000	English Text	Hate-speech, Offensive language, Neither	Kaggle

3.2. Roman Urdu Hate-speech and Offensive Language Dataset

This is the dataset generated by Dr. Asim Karim's project [15] on hate speech detection in Roman Urdu. A novel lexicon of hateful words (derogatory terms, abuses, slurs, religious and sexist abusive words etc.) in Roman Urdu was constructed, after carrying out online search for such words and conducting interviews. 50, 000 tweets were gathered, with 10, 000 tweets randomly selected for manual labeling by annotators. The following labels were used:

1. Abusive/Offensive
2. Sexist
3. Profane
4. Religious Hate
5. Normal

Domain-specific Roman Urdu embeddings were trained on 4.7 million tweets, and each word represented as a 300-dimensional vector.

3.3. Final Dataset

3.3.1 Labels

We mainly took inspiration from Dr. Asim Karim's project [15] and the labels used there.

From those labels, we chose the following:

a. Sexist → Language used to express hatred towards a targeted individual or group based on gender or sexual orientation [16].
b. Religious Hate → Language used to express hatred towards a targeted individual or group based on their religious beliefs or lack of any religious beliefs [17].
c. Profane → The use of vulgar, foul or obscene lanLablel Tweet Count Abusive/Offensive 2, 402 Sexism 839 Religious Hate 782 Profane 640 Normal 5, 349 Total 10, 012 Table 2: Tweet counts with respect to labels for fine grained classification task gauge without an intended target [18].
d. Normal → This contains text that does not fall into the above categories

When choosing our final labels, our prime concern was to ensure there was no overlap or correlation between any 2 labels. This criteria was employed when choosing the labels. Since there was a lot of overlap of the label Abusive/Offensive with the label profane, we chose to keep profane only. In Dr. Asim's project, the former included vulgar language towards an intended target while the latter was the same but without a target. We chose profane as the label, regardless of the involvement of a target. We defined profane to include vulgar or obscene language with or without an intended target

3.3.2 Data Collection

We started off by using Twitter API to collect tweets. However, the Problem with Twitter API was that we were getting a maximum of 300 tweets in one whole round. Therefore, we switched to using TWINT to scrape tweets from twitter, and used the negative words lexicon from Kaggle.

Further, we built a program that randomly chose a word from that lexicon and searched for tweets containing that particular word. In order to increase the number of tweets for each category/class, we searched using words on our own as well.

Dataset Phase I. The 1st dataset we generated contained around 2700 tweets that we manually labelled. The problem we encountered was that of a huge class imbalance. A very large proportion – in fact almost all of the tweets belonged to the 'Normal' category (no hate speech), with only a few that fit the other 3 categories. Another challenge was that a large number of tweets were in languages other than Urdu, or contained just a single word, so all such tweets had to be discarded/not considered. After removing all such tweets, we came up with a dataset of around 1300 tweets.

Dataset Phase II. The 2nd dataset we generated contained around 2200 tweets that we manually labelled out of a total of 4000 extracted tweets. This time since we searched/scraped using specific words, we were partially able to counter class imbalance. However, the same issue remained regarding the language and size of tweets after data cleaning and filtering.

Dataset Phase III. Our 3rd phase consisted of 2 steps

First, from Sir Asim's dataset, we removed all the tweets with the 'Abusive/Offensive' label as we did not incorporate this label in our project. Hence, after filtering out all such tweets, we were left with around 6,900 tweets.

Second, we further extracted 3,000 tweets from Twitter and after discarding tweets that were not fit for labeling, similar to phases 1 and 2, we ended up with around 1400 tweets. Adding these to our dataset from the 1st 2 phases, our total manually labeled dataset summed up to roughly 5000 tweets.

Now, we merged these 5000 tweets we manually labeled, with Sir Asim's filtered dataset of size 6,900. Therefore, our final dataset summed up to a size of approximately 12,000 tweets.

3.3.3 Annotation

Each tweet was manually labeled as belonging to one of the 4 classes defined earlier. Tweets that were based on a single word, or were in a language other than Roman Urdu, were discarded and not labeled. Tweets that raised ambiguity as to which class they belonged to were set aside for

later review and discussion. Each tweet was labeled by all 3 annotators and the majority label was selected as the final label.

3.3.4 Evaluation protocol.

To evaluate the reliability of our annotation we used the Cohen's Keppa metric. Cohen's kappa would be used to measure the agreement between multiple annotators who each tweet into one of the 4 selected mutually exclusive labels. This would give us a quantitative measure of reliability for two or more annotators that are labeling the same tweets, corrected for how often that the annotators may agree by chance. For our dataset we got an agreement score of 99.503% and a Cohen's Keppa of 0.3218. This value signifies that our dataset annotation was indeed 'fair'

Chapter 4

4. METHODOLOGY

4.1. DATASET PRE-PROCESSING

We considered 3 existing data embedding techniques for our models. These included a unigram bag-of-words, word tokenization and lastly using Dr. Haroon's Roman Urdu embeddings.

Bag of Words. We represented each tweet as a unigram bag-of-words representation which gave us an unordered set of words with their position ignored, keeping only their frequency in the tweet. This helped us convert our textual input into numerical data. Subsequently, we applied a bag of word representation to each of our english and roman urdu datasets before running it through our models.

Word Tokenization. For this technique we compiled our entire dataset to assign a token to each unique word. Then each word of a sentence is replaced with their assigned token value using a tokenizer. To pass this data as an input to our models all sequences in the entire dataset needed to have the same length. One solution for this was to use the length of the longest sequence in our dataset but this would result in memory wastage especially on a large dataset. Hence, we reached a compromise by using a sequence length which covers most sequences in our datasets while subsequently, truncating longer sequences and padding shorter sequences. Resultantly our chosen sequence length covered more that 92% of the data for each of our dataset. Moreover, we pre-padded our dataset so that our padded zero values don't later confuse our machine learning models. Finally, we embedded our tokens using our keras model and passed the resultant matrix as an input to the model itself.

Roman Urdu Embeddings. Our final preprocessing technique involved embedding our tweets using Dr. Haroon's Roman Urdu embeddings before passing them onto our model. These embeddings were trained over 4.2 million tweets and represented each Roman Urdu word as a feature vector of 300 dimensions.

Furthermore, we used a uniform 80:20 train-test stratified split for each of our models to achieve a similar class ratio for both test and train data.

4.2. MODELS

We aimed to experiment with multiple machine learning models in order to reach an optimum model for our given solution. For this we used basic SKLearn models as well as sequence learning models to compare a contextual model with a non contextual one.

SKLearn Models. We used Python's Scikit-learn library to train general machine learning models on both our pre-existing and our final dataset. The algorithms we used for our initial dataset gave us an estimate for the optimum algorithm required for our problem. We used algorithms such as Logistic Regression, Naive Bayes, Multinomial Naive Bayes, K- Nearest Neighbor and Decision Trees.

Recurrent Neural Network Models. To counter our problem using sequence based learning algorithms we used keras' in-built Recurring Neural Network(RNN) model. RNN has the ability to use its internal state (memory) to process sequenced inputs which can help tackle problems such as ours using a more contextual approach. We experimented with 3 different variants RNN models on our dataset. The difference in these models was the building block of the model. The three building blocks are as follows:

1) Gated Recurrent Unit (GRU)
2) Long Short-Term Memory (LSTM)
3) Bi-directional Long Short-Term Memory (LSTM)

These variants would allow us to compare the rather clunky LSTM, the somewhat simpler GRU and a comparatively complex Bi-directional LSTM for the optimum contextual model.

Model Evaluation Metric. To evaluate the performance of our model we decided to use the accuracy and the weighted average score for the 4-label classification of our model on the test data.

Chapter 5

5. Implementation and Evaluation

5.1. SK Learn Models

To start off we trained and tested our sklearn models on the pre-existing datasets for both binary and multi-class, english and roman urdu datasets. We experimented with 2 different embeddings for our models: bag-of-words and word tokenization.

5.1.1 Bag of Words

We embedded each of our datasets using the unigram bag-of-words representation before feeding it to our model.

Binary Class English Tweets. We trained separate Logistic Regression, Multinomial Naive Bayes and KNN models and our best performing model out of these models was the Logistic Regression model which gave us an accuracy of 0.95 with a weighted average f1 score of 0.95 as well. However, our other models also performed almost as good as the Logistic Regression model as shown in Table 2.

Table 2 BAG-OF-WORDS WITH BINARY CLASS ENGLISH TWEETS

Model	Accuracy	Weighted Average f1-score
Logistic Regression	0.95	0.95
KNN (k=3)	0.94	0.93
Multinomial Naive Bayes	0.94	0.94

Binary Class Roman Urdu Tweets. Next we used our two-class Roman Urdu dataset to train our sklearn models. We trained the same sklearn models as the ones we used on the previous dataset. Similarly, as shown by Table 3 our best performing algorithm was Logistic Regression, giving us an accuracy of 0.99 and a weighted average f1 score of 0.98 while our second best performing model was Multinomial Naive Bayes.

Table 5.1 BAG-OF-WORDS WITH BINARY CLASS (NEUTRAL/HOSTILE) ROMAN URDU TWEETS

Model	Accuracy	Weighted Average f1-score
Logistic Regression	0.99	0.98
KNN (k=3)	0.84	0.84
Multinomial Naive Bayes	0.92	0.92

Table 5.2 BAG-OF-WORDS WITH BINARY CLASS (HATE/OFFENSIVE) ROMAN URDU TWEETS

Model	Accuracy	Weighted Average f1-score
Logistic Regression	0.99	0.99
KNN (k=3)	0.87	0.87
Multinomial Naive Bayes	0.94	0.94

Multi-Class English Tweets. Our best performing model yet again was Logistic Regression which gave us an accuracy of 0.79 and a weighted average f1 score of 0.76 as evident by Table 4.

Table 5.3 BAG-OF-WORDS WITH MULTI-CLASS ENGLISH TWEETS

Model	Accuracy	Weighted Average f1-score
Logistic Regression	0.91	0.91
KNN (k=3)	0.82	0.83
KNN (k=5)	0.82	0.83
KNN (k=7)	0.82	0.82
KNN (k=9)	0.81	0.81
Multinomial Naive Bayes	0.87	0.87

Multi-Class Roman Urdu Tweets. Finally, we trained our sklearn models on the Roman Urdu dataset with 3-class classification. Interestingly, this time we had 2 algorithms tied for the best performance; Logistic Regression and Multinomial Naive Bayes both gave us an accuracy of 0.66 but Logistic Regression had a slightly higher weighted average f1 score of 0.65 compared to 0.64 of its counterpart shown in Table 5.

Table 5.4 BAG-OF-WORDS WITH MULTI-CLASS ROMAN URDU TWEETS

Model	Accuracy	Weighted Average f1-score
Logistic Regression	0.66	0.65
KNN (k=3)	0.52	0.44
KNN (k=5)	0.56	0.44
KNN (k=7)	0.55	0.42
KNN (k=9)	0.57	0.40
Multinomial Naive Bayes	0.66	0.64

5.1.2 Roman Urdu Embedding

Our next feature vector representation involved using Dr. Haroon's Roman Urdu Embeddings on our dataset. Unfortunately, Multinomial Naive Bayes does not accept the negative values used in the embedding hence we continued with Logistic Regression and KNN only. Subsequently, we achieved an accuracy of 0.72 and a weighted average f1 score of 0.70 with Logistic Regression as it was also our best performing model.

Table 5.5 ROMAN URDU EMBEDDINGS WITH OUR FINAL DATASET

Model	Accuracy	Weighted Average f1-score
Logistic Regression	0.72	0.70
KNN (k=3)	0.67	0.64
KNN (k=5)	0.68	0.64
KNN (k=7)	0.70	0.64
KNN (k=9)	0.70	0.63

5.2. Recurrent Neural Network Models

For our sequential based RNN models we used word tokenization as our feature vector representation and ran each of our models for 5 epochs. Moreover, we implemented our RNN models using keras. Each model embedded our input into a vector of size 64 before passing it on to the other layers. Additionally, we took a batch size of 64, and a validation split of 0.05 while taking the categorical cross entropy as our loss function.

Gated Recurrent Unit. For our first RNN model we passed our embedded layers through three GRU layers. Figure 1 shows the internal structure of our model. After the model was trained we passed our test dataset to evaluate the performance; this model gave us an accuracy of 0.77 and a weighted f1 score of 0.72. The training evaluation is shown in Figure 2.

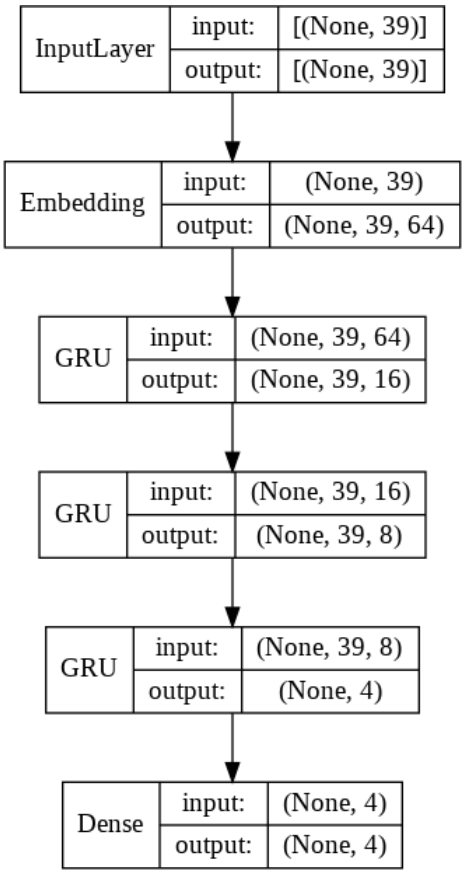


Fig. 5.1 RNN model with GRU layers.

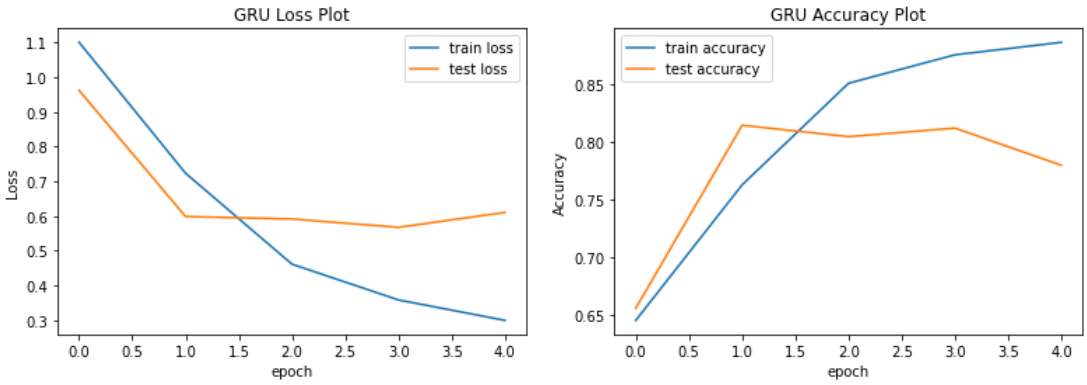


Fig. 5.2 .RNN model with GRU layers loss and accuracy plots

Long Short-Term Memory. Next, we implemented a RNN model consisting of LTSM as its basic building block. Similarly we embedded our input before passing on to the other LTSM layer as shown in Figure 3. After training our model (Figure 4) we got an accuracy of 0.75 and an f1-score of 0.77.

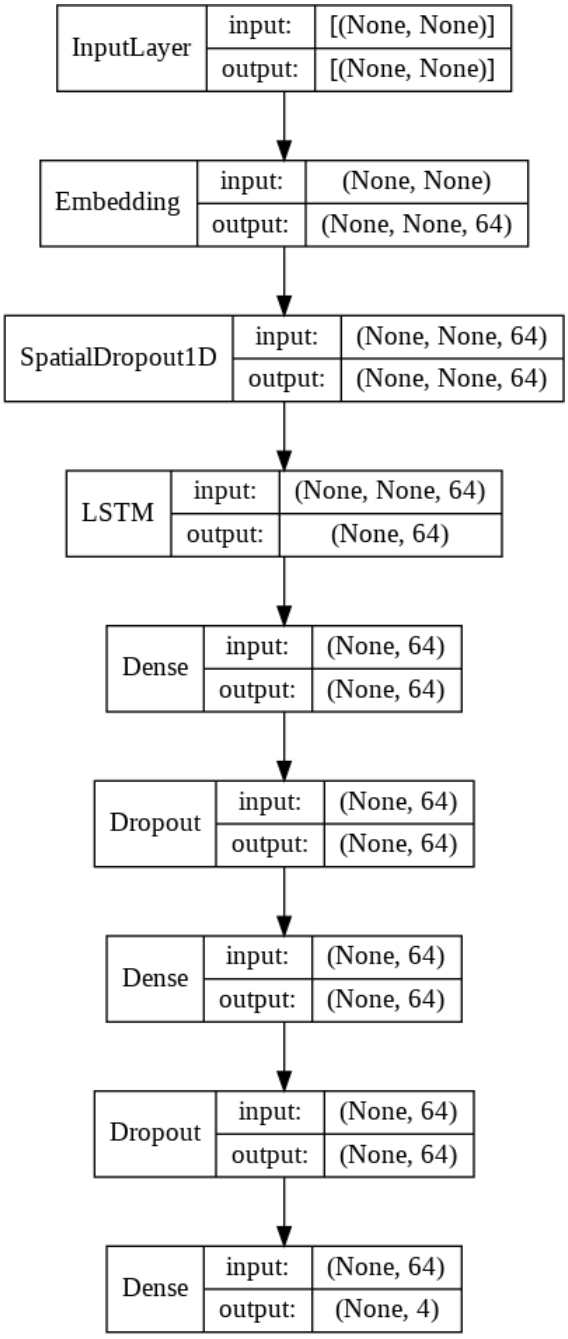


Fig. 5.3 RNN model with LSTM layer.

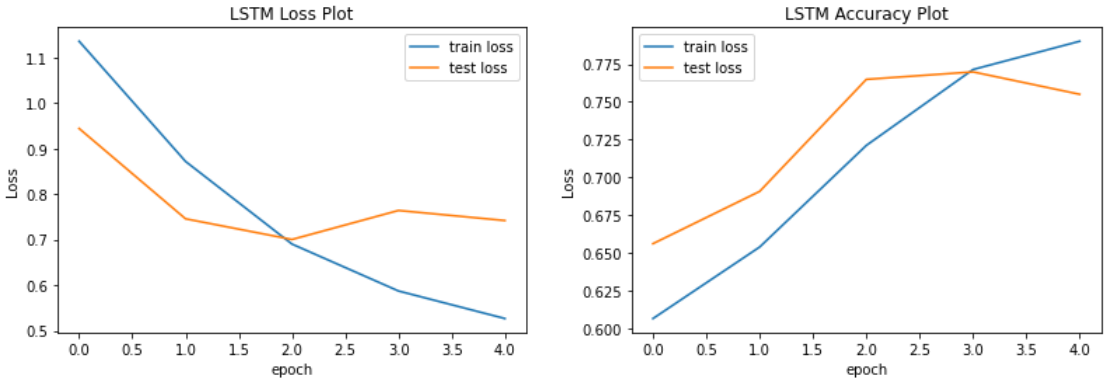


Fig. 5.4 RNN model with LSTM layer loss and accuracy plots

Bi-directional Long Short-Term Memory. Our final RNN model consisted of a bidirectional LSTM as its core building block. We trained our model by passing the feature vector through a bidirectional LSTM layer among other layers (Figure 5). After training for an epoch of 5 we got an accuracy of 0.77 as shown in Figure 6. This model also gave us our highest f1-score of 0.78

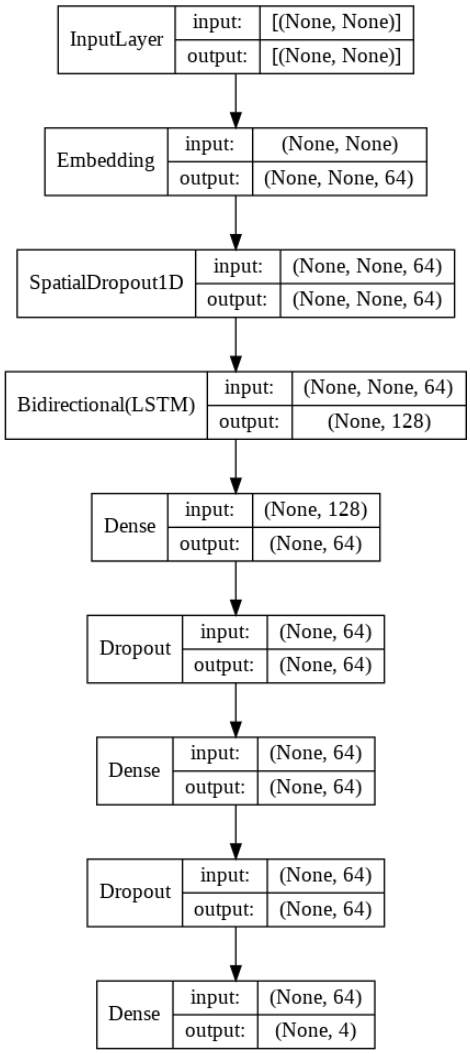


Fig. 5.5 RNN model with bidirectional LSTM layer.

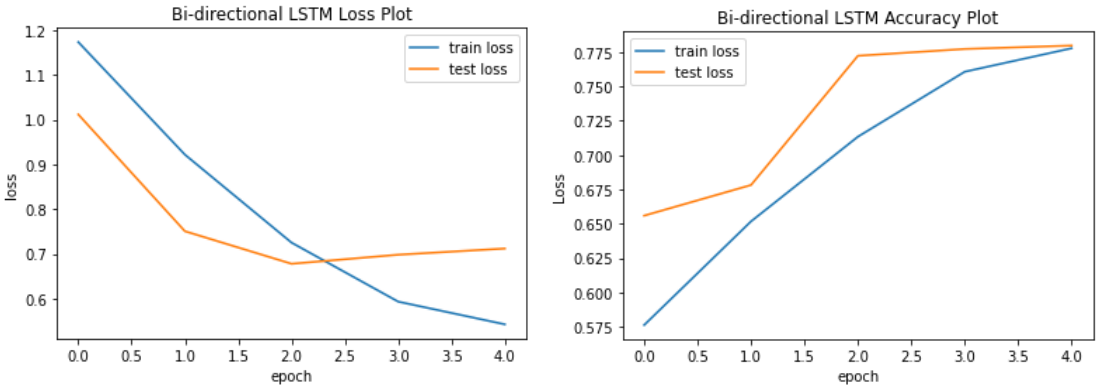


Fig. 5.6 RNN model with bidirectional LSTM layers loss and accuracy plots

5.3. Result Analysis

The Recurrent neural networks clearly perform better than the sklearn models making use of their sequential learning - learning the word context in the sentence rather than the word count. Additionally GRU makes use of its simpler structure and classifies our given tweets with the joint highest accuracy as most of the tweets in the dataset do not contain large word sequences. LSTM on the other hand performs better with longer tweets since it maintains information in memory for long periods of time. Lastly, the bidirectional LSTM outperforms our other models due to the fact that it is able to understand the context of the input better because of its ability to preserve information from both past and future in any point.

Chapter 6

6. Conclusion and Future Work

Concludingly, we were able to successfully extract a considerable number of raw tweets from Twitter, as well as manually annotate them. Subsequently, we generated a large and reliable dataset. Experiments revealed that the best performance was achieved by the bidirectional LSTM, providing an accuracy of 77% and F1-score of 0.78.

Within the scope of this project we can further improve our models to achieve better scores while also trying to collect more sample tweets for better model training. In order to deal with the issue of bias, multiclass classification could be looked into, with multiple labels being assigned to a single tweet. Further, an improvement in the Roman Urdu embeddings could be attempted at. The dataset could be expanded to around 100,000 tweets, and an efficient way to counter the existing class imbalance could be explored.

Future work in this field can focus on identifying, with more precision, the different types of hate speech sentences, for example, investigating whether a sentence is a threat or a plain remark, exploring the potential a sentence or comment has to incite violence. The relationships between user characteristics such as geographical location, time of posting, global or regional events around the time the comment or tweet was posted, could be studied along with their effects on the content posted.

REFERENCES

[1] Watanabe, Hajime, Mondher Bouazizi, and Tomoaki Ohtsuki. "Hate speech on twitter: A pragmatic approach to collect hateful and offensive expressions and perform hate speech detection." *IEEE access* 6 (2018): 13825-13835.

[2] Gaydhani, Aditya, et al. "Detecting hate speech and offensive language on twitter using machine learning: An n-gram and tfidf based approach." arXiv preprint arXiv:1809.08651 (2018).

[3] Pitsilis, Georgios K., Heri Ramampiaro, and Helge Langseth. "Effective hate-speech detection in Twitter data using recurrent neural networks." Applied Intelligence 48.12 (2018): 4730-4742.

[4] Waseem, Zeerak, and Dirk Hovy. "Hateful symbols or hateful people? predictive features for hate speech detection on twitter." *Proceedings of the NAACL student research workshop*. 2016.

[5] Masud, Sarah, et al. "Hate is the New Infodemic: A Topic-aware Modeling of Hate Speech Diffusion on Twitter." *arXiv preprint arXiv:2010.04377* (2020).

[6] Alatawi, Hind Saleh, Areej Maatog Alhothali, and Kawthar Mustafa Moria. "Detecting White Supremacist Hate Speech using Domain Specific Word Embedding with Deep Learning and BERT." *arXiv preprint arXiv:2010.00357* (2020).

[7] J. M. Schneider, R. Roller, P. Bourgonje, S. Hegele, and G. Rehm, ‘‘Towards the automatic classification of offensive language and related phenomena in German tweets,’’ in Proc. 14th Conf. Natural Lang. Process. (Konvens), 2018, p. 95.

[8] Pereira-Kohatsu, Juan Carlos, et al. "Detecting and monitoring hate speech in Twitter." *Sensors* 19.21 (2019): 4654.

[9] M. O. Ibrohim and I. Budi, ‘‘A dataset and preliminaries study for abusive language detection in Indonesian social media,’’ Procedia Comput. Sci., vol. 135, pp. 222–229, Jan. 2018.

[10] R. Pelle, C. Alcântara, and V. P. Moreira, ‘‘A classifier ensemble for offensive text detection,’’ in Proc. 24th Brazilian Symp. Multimedia Web, 2018, pp. 237–243.

[11] G. I. Sigurbergsson and L. Derczynski, ‘‘Offensive language and hate speech detection for Danish,’’ Aug. 2019, arXiv:1908.04531. [Online]. Available: https://arxiv.org/abs/1908.04531

[12] A. Alakrot, L. Murray, and N. S. Nikolov, ‘‘Towards accurate detection of offensive language in online communication in Arabic,’’ Procedia Comput. Sci., vol. 142, pp. 315–320, Jan. 2018.

[13] Akhter, Muhammad Pervez, et al. "Automatic detection of offensive language for urdu and roman urdu." *IEEE Access* 8 (2020): 91213-91226.

[14] Mehmood, Khawar, et al. "Sentiment analysis for a resource poor language—Roman urdu." *ACM Transactions on Asian and Low-Resource Language Information Processing (TALLIP)* 19.1 (2019): 1-15.

[15] Rizwan, Hammad, Muhammad Haroon Shakeel, and Asim Karim. "Hate-speech and offensive language detection in roman Urdu." *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2020.

[16] Zeerak Waseem and Dirk Hovy. 2016. Hateful symbols or hateful people? predictive features for hate speech detection on twitter. In Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL): Student Research Workshop, pages 88–93.

[17] William Warner and Julia Hirschberg. 2012. Detecting hate speech on the world wide web. In Proceedings of the 2nd Workshop on Language in Social Media, pages 19–26.

[18] Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated hate speech detection and the problem of offensive language. In 11th International AAAI Conference on Web and Social Media, (ICWSM), pages 512–515.