Presentation On

DETECTION OF HATE SPEECH AND CYBER AGGRESSION ON SOCIAL MEDIA

Presented by

Syeda Zarif Afrin (10200116020)
Pravash Ranjan Nayak (10200116043)
Pankaj Kumar Mundhra (10200116044)
Mayukh Sengupta (10200116047)

Under the guidance of

Dr. Kousik Dasgupta
Assistant Professor
Computer Science and Engineering
Kalyani Government Engineering College

Kalyani Government Engineering College
(Affiliated to Maulana Abul Kalam Azad University of Technology, West Bengal)
Kalyani - 741235, Nadia, WB

Content

1. Introduction
2. Existing Work
3. Proposed Work
4. Preprocessing
5. Model 1:Feature Extraction using NLP Techniques and applying Linear Models
6. Model 2:Fully Connected Feed Forward Neural Network
7. Conclusion
8. Future Work

Introduction

- An exponential increase in the use of web by people of various cultures and academic background has made toxic online content become a significant issue in today's world.
- The goal of this project is to look at how Machine Learning and Deep Learning applies in detecting hate speech. The Artificial Neural Network classifier, used in the project assigns each tweet to one of the categories of a Twitter dataset: hate/offensive language or neither.
- This also includes use of Natural Language Processing and n-gram feature extraction to convert raw text into somewhat meaningful vectors.
- The performance of this model has been tested using the accuracy.

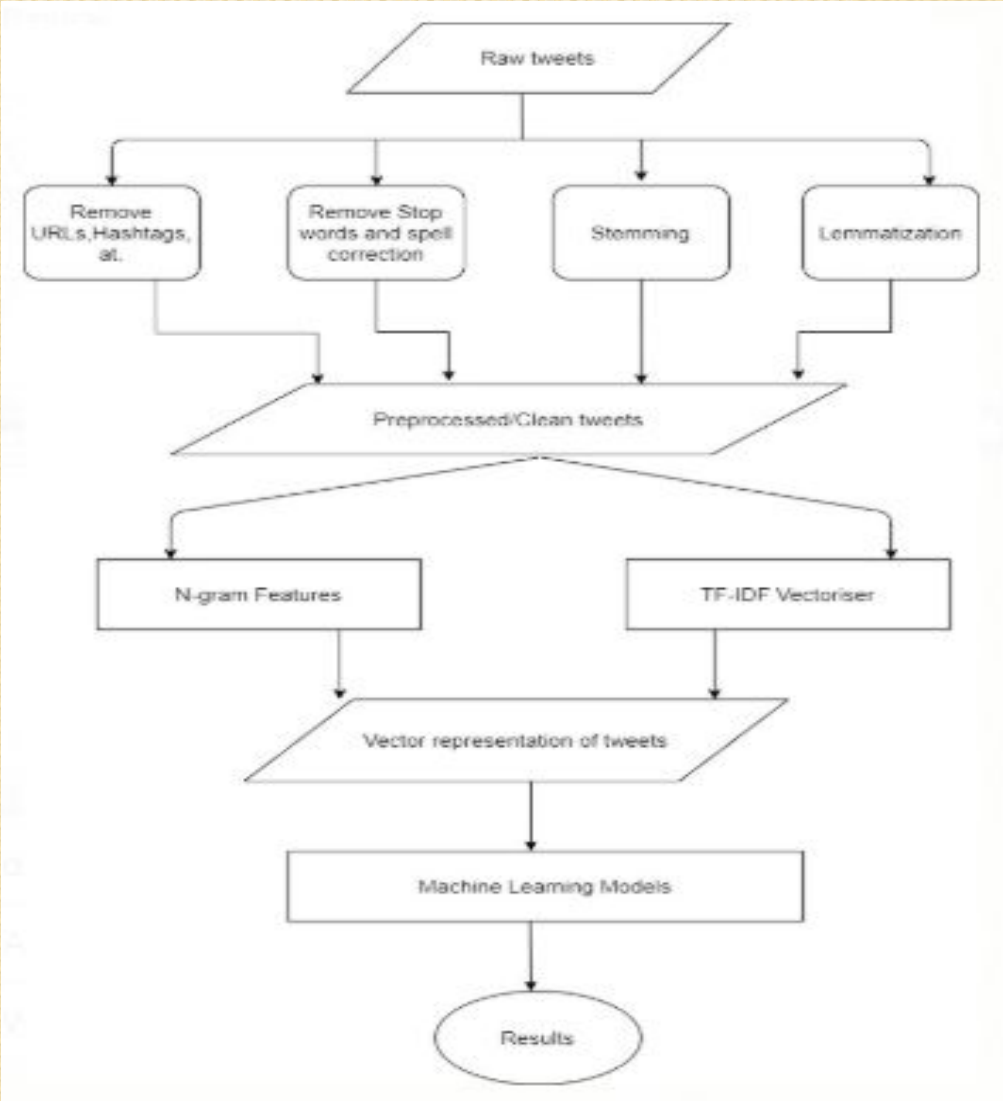
Existing Work

Some of the examples of existing work are:

- The European Union sponsored project PRINCIP has used support vector machines using a bag-of-words approach to classify Web pages containing racist text [Greevy and Smeaton 2004].
- Linguistic features such as parts-of-speech has also been used in hate speech detection problem; these approaches consist in detecting the category of the word, for instance, personal pronoun, Verb non-3rd person singular present form, Adjectives, Determiners, Verb base forms (VB). There have been several studies on sentiment-based methods to detect abusive language published in the last few years. [Department of Computer Engineering, Maharashtra Institute of Technology, Pune Pune, India]
- There are also studies which typically employ semantic content analysis techniques built on Natural Language Processing (NLP) and Machine Learning (ML) methods, both of which are core pillars of the Semantic Web research. The task typically involves classifying textual content into non-hate or hateful, in which case it may also identify the types of the hate speech. [Information School, University of Sheffield, Regent Court, 211 Portobello, Sheffield, S1 4DP, UK]
- Logistic regression with L1 regularization was used reduce the dimensionality of the data. And then tested a variety of models that have been used in prior work: logistic regression, na¨ıve Bayes, decision trees, random forests, and linear SVMs. [Department of Sociology, Cornell University, Ithaca, NY, USA]

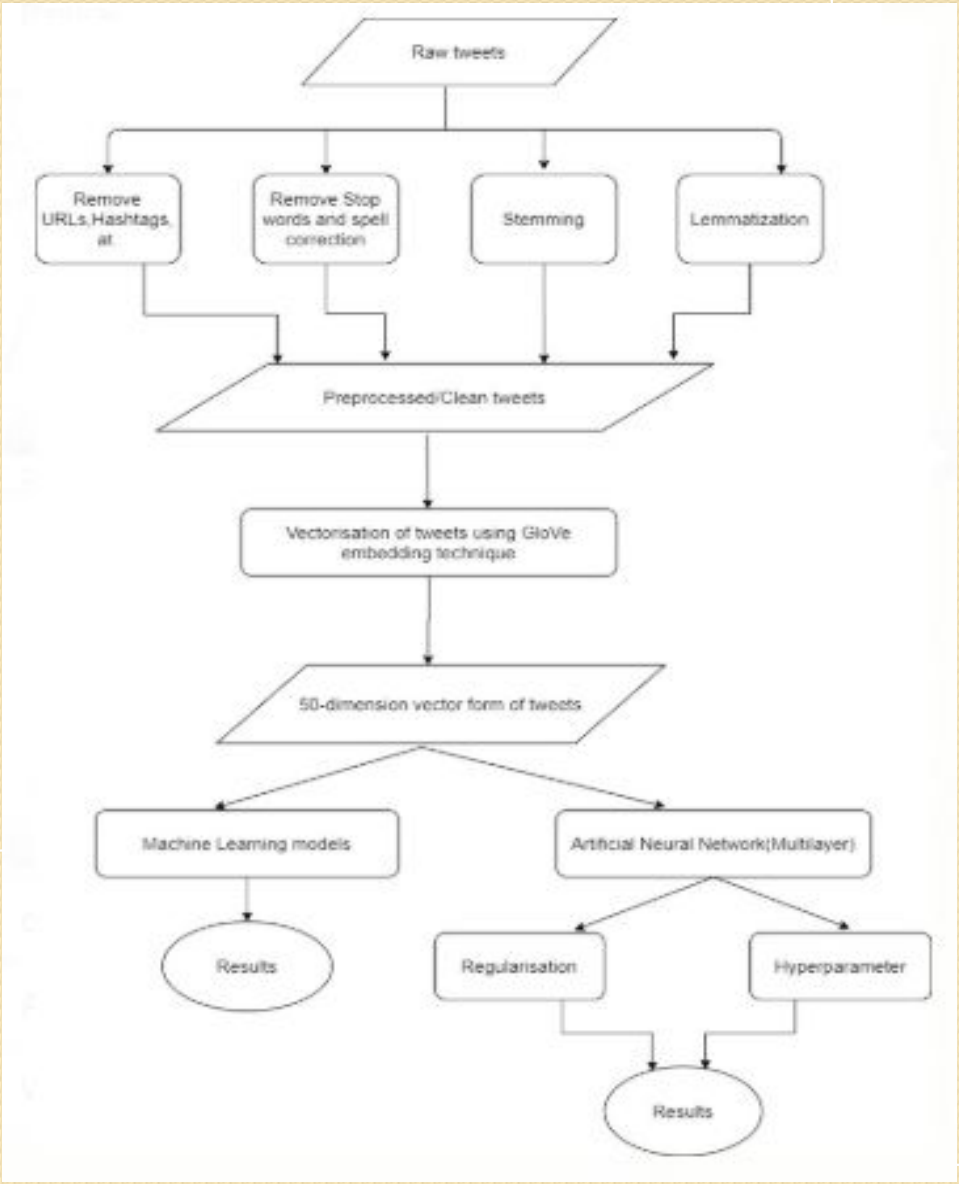
Proposed Work

Model 1:Feature Extraction using NLP Techniques and applying Linear Models



- N gram feature and TFIDF vectorizer is used to vectorize each tweets.
- Linear Machine Learning model like Logistic Regression is applied to it.
- Better results are achieved using regularisation and hyper parameterization.

Model II: Fully Connected Feedforward Neural Network



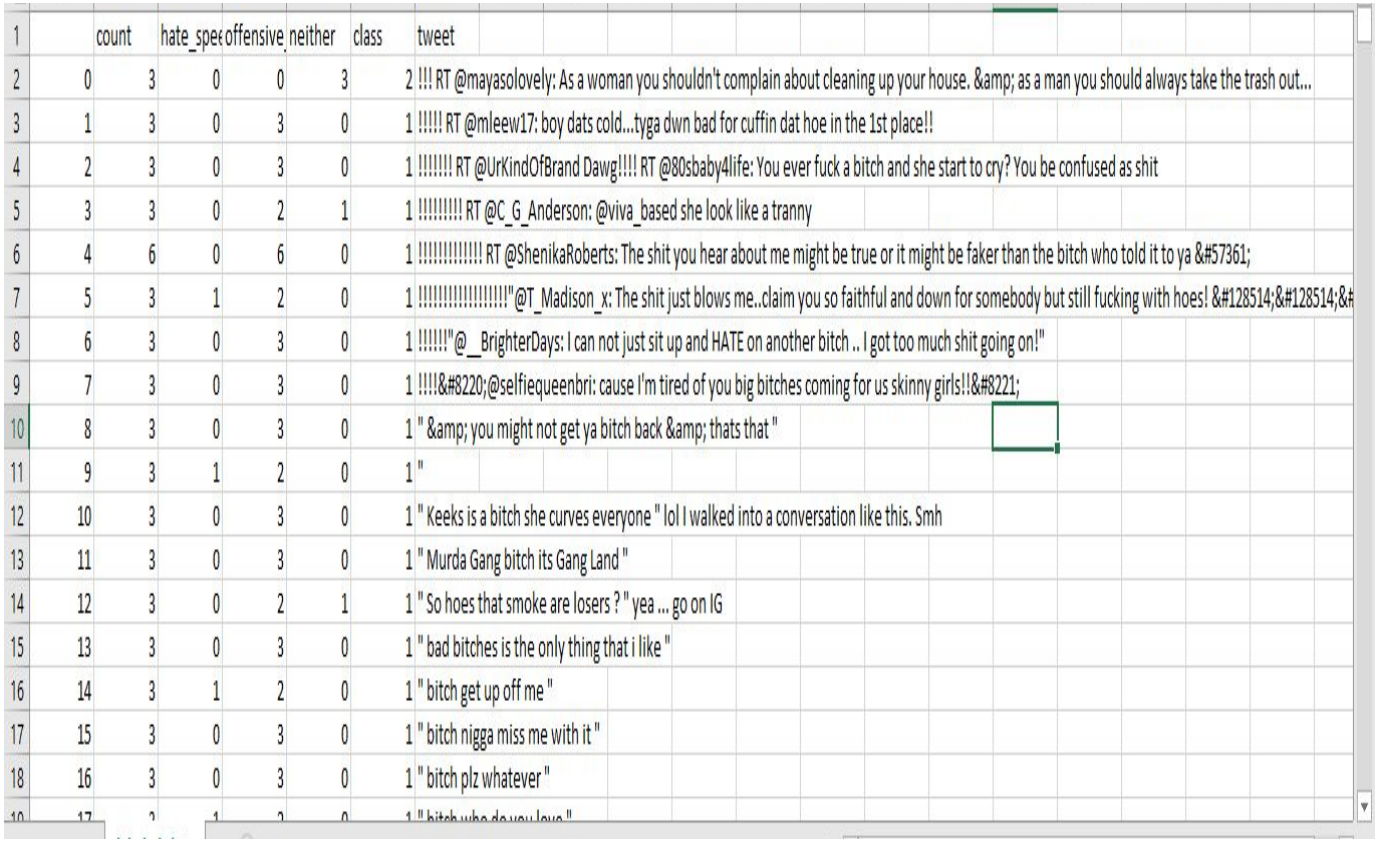
- GloVe vectorization is used to convert tweet to 50 dimensional vectors.
- Linear Machine Learning models are applied.
- 2 types of network configurations are applied in Multi Layered Perceptron.
- Regularization and Hyper parameterization is used to get better results.

Dataset

The dataset contains 14163 tweets.

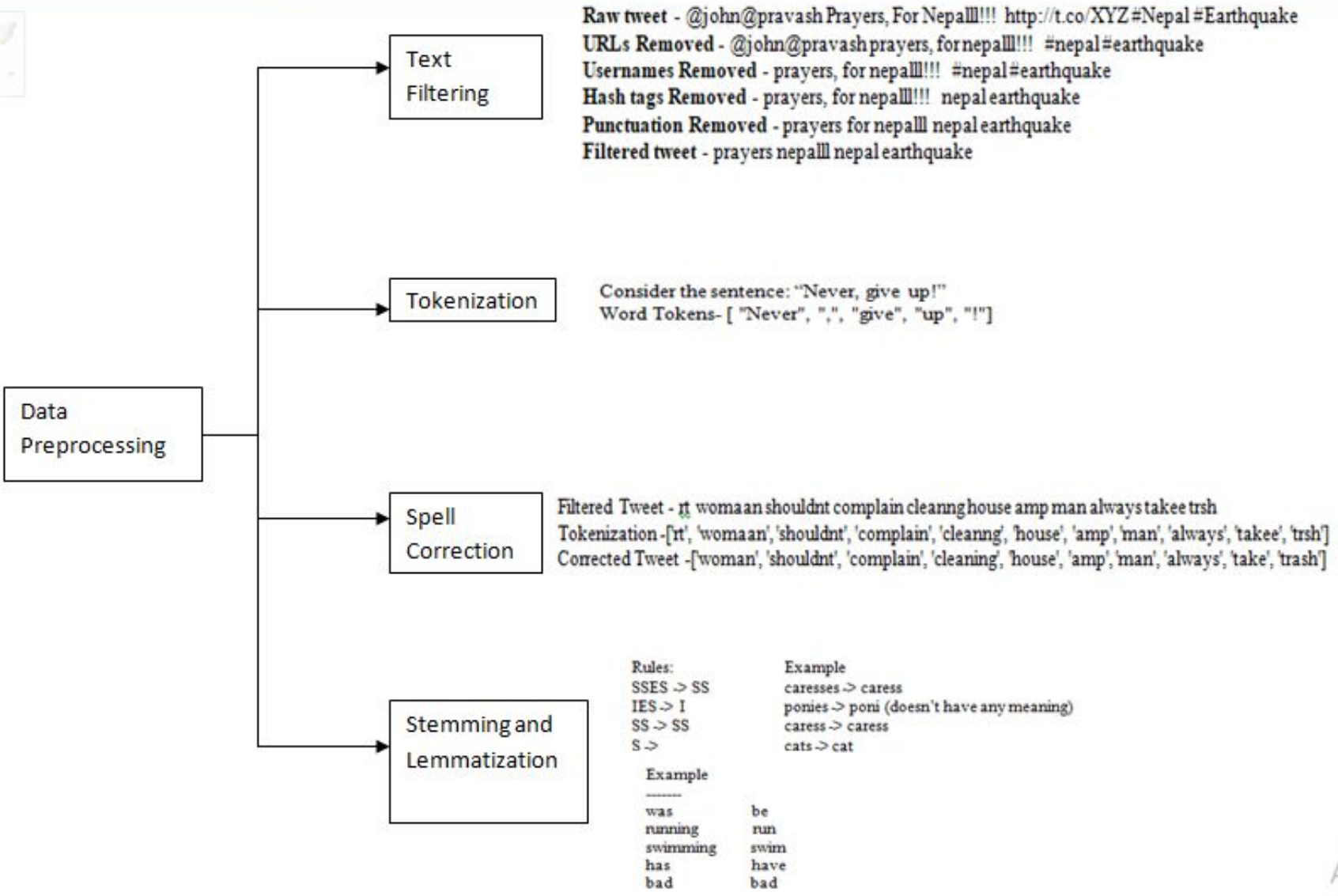
Hate tweets: 10000

Non hate tweets: 4163



		count	hate_spe	offensive	neither	class	tweet
2	0	3	0	0	3	2	!!! RT @mayasolovely: As a woman you shouldn't complain about cleaning up your house. & as a man you should always take the trash out...
3	1	3	0	3	0	1	!!!!! RT @mleew17: boy dats cold...tyga dwn bad for cuffin dat hoe in the 1st place!!
4	2	3	0	3	0	1	!!!!!!! RT @UrKindOfBrand Dawg!!!! RT @80sbaby4life: You ever fuck a bitch and she start to cry? You be confused as shit
5	3	3	0	2	1	1	!!!!!!!!! RT @C_G_Anderson: @viva_based she look like a tranny
6	4	6	0	6	0	1	!!!!!!!!!!!! RT @ShenikaRoberts: The shit you hear about me might be true or it might be faker than the bitch who told it to ya 
7	5	3	1	2	0	1	!!!!!!!!!!!!!!!!!"@T_Madison_x: The shit just blows me..claim you so faithful and down for somebody but still fucking with hoes! 😂😂&#
8	6	3	0	3	0	1	!!!!!!"@__BrighterDays: I can not just sit up and HATE on another bitch .. I got too much shit going on!"
9	7	3	0	3	0	1	!!!!“@selfiequeenbri: cause I'm tired of you big bitches coming for us skinny girls!!”
10	8	3	0	3	0	1	" & you might not get ya bitch back & thats that "
11	9	3	1	2	0	1	"
12	10	3	0	3	0	1	" Keeks is a bitch she curves everyone " lol I walked into a conversation like this. Smh
13	11	3	0	3	0	1	" Murda Gang bitch its Gang Land "
14	12	3	0	2	1	1	" So hoes that smoke are losers ? " yea ... go on IG
15	13	3	0	3	0	1	" bad bitches is the only thing that i like "
16	14	3	1	2	0	1	" bitch get up off me "
17	15	3	0	3	0	1	" bitch nigga miss me with it "
18	16	3	0	3	0	1	" bitch plz whatever "
19	17	3	1	2	0	1	" bitch who do you love "

Data Preprocessing



Model 1:Feature Extraction using NLP Techniques and applying Linear Models

Feature extraction using n grams and TFIDF vectorizer

Text Corpus Example: [*'how are you', 'good to see you"']*

Unigrams and Bigrams: [*'are', 'are you', 'good', 'good to', 'how', 'how are', 'see', 'see you', 'to', 'to see']*

Term Frequency Inverse Document Frequency Vectorizer(TFIDF)

$$TFIDF\ score\ for\ term\ i\ in\ document\ j = TF(i,j) * IDF(i)$$

where

$$IDF = Inverse\ Document\ Frequency$$

$$TF = Term\ Frequency$$

$$TF(i,j) = \frac{\text{Term i frequency in document j}}{\text{Total words in document j}}$$

$$IDF(i) = \log_2 \left(\frac{\text{Total documents}}{\text{documents with term i}}\right)$$

and

$$t = Term$$

$$j = Document$$

Example of TFIDF vectorization

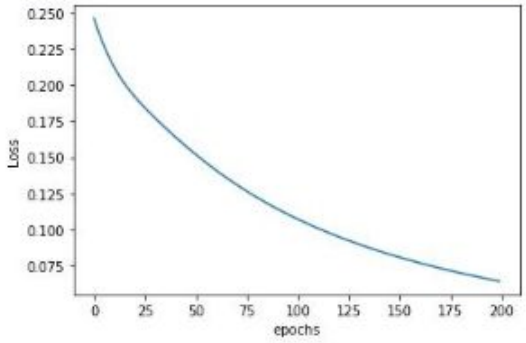
	are	are you	good	good to	how	how are	see	see you	to	to see	you
0	0.471078	0.471078	0.000000	0.000000	0.471078	0.471078	0.000000	0.000000	0.000000	0.000000	0.335176
1	0.000000	0.000000	0.392044	0.392044	0.000000	0.000000	0.392044	0.392044	0.392044	0.392044	0.278943

Logistic Regression

Here we have applied logistic regression which is a linear model on this TFIDF vector of tweets.

LOSS CURVE

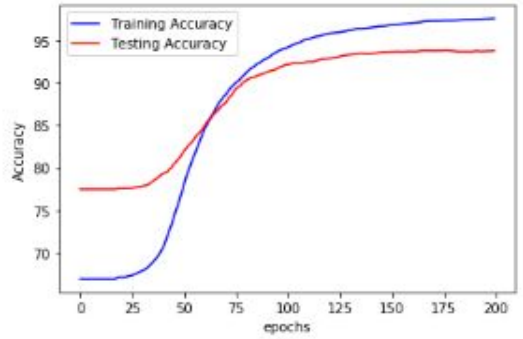
Model: Logistic Regression , Loss fn: MSE , Optimizer: Adam , LR: 0.0001 , Epochs: 200 , Batch_size= 30



This depicts the rate of change of loss with each epoch for the given model

ACCURACY vs EPOCHS

Model: Logistic Regression , Loss fn: MSE , Optimizer: Adam , LR: 0.0001 , Epochs: 200 , Batch_size= 30



This depicts the change of Train and Test set accuracy with each epoch for the given model

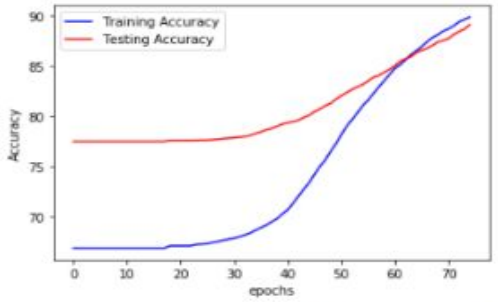
TRAIN-TEST ACCURACY

```
In [36]: #Train Accuracy
         ((fn(X_train_tensor.float())>0.5).type(torch.int)==y_train_tensor).sum().item()/y_train.shape[0]*100

Out[36]: 97.63333333333334

In [37]: #Test Accuracy
         ((fn(X_test_tensor.float())>0.5).type(torch.int)==y_test_tensor).sum().item()/y_test.shape[0]*100

Out[37]: 93.82142165407708
```

Logistic Regression with Regularisation

EARLY STOPPING REGULARIZATION TO PREVENT OVERFITTING

```
Model: Logistic Regression , Loss fn: MSE , Optimizer: Adam , LR: 0.0001 , Epochs: 75 , Batch_size= 30
```



FINAL TRAIN-TEST ACCURACY

```
In [79]: #Train Accuracy
         ((fn(X_train_tensor.float())>0.5).type(torch.int)==y_train_tensor).sum().item()/y_train.shape[0]*100
Out[79]: 89.83333333333333

In [80]: #Test Accuracy
         ((fn(X_test_tensor.float())>0.5).type(torch.int)==y_test_tensor).sum().item()/y_test.shape[0]*100
Out[80]: 89.07611853573503
```

PERFORMANCE ON OUT OF VOCABULARY SENTENCES

```
"'cuteie pie" - Hate
"all jews are swine" - Hate
"you are a friendly person" - Hate
"shut up nigger" - Hate
"black people deserve to burn in hell" - Hate
"what are you looking at , dumb fuck !" - Hate
"good people are found everywhere" - Hate
```

- Overfitting was prevented by Regularization Technique called Early Stopping
- The model then achieved a reasonably higher accuracy doing well on both Train and Test datasets
- Due to the drawbacks of TF-IDF vectors , the model failed to classify sentences that were outside the corpus on which it was trained on.

Model II: Fully Connected Feedforward Neural Network

Feature extraction using GloVe embeddings

GloVe embedding provides us with a vectorisation of a particular word by giving an n-dimensional array, which can be of length 50,100,200 and 300. For our purpose we have used 50 dimensions and rest are up to experimentation purposes.

Example:

"Hello"=[-0.38497,0.80092,0.064106,-0.28355,-0.026759,-0.34532,-0.64253,-0.11729,-0.33257,0.55243,-0.087813,0.9035,0.47102,0.56657,0.6985,-0.35229,-0.86542,0.90573,0.03576,-0.071705,-0.12327,0.54923,0.47005,0.35572,1.2611,-0.67581,-0.94983,0.68666,0.3871,-1.3492,0.63512,0.46416,-0.48814,0.83827,-0.9246,-0.33722,0.53741,-1.0616,-0.081403,-0.67111,0.30923,-0.3923,-0.55002,-0.68827,0.58049,-0.11626,0.013139,-0.57654,0.048833,0.67204]

GloVe vectors with ML models

	MODELNAME	ACCURACY	PRECISION	RECALL	AUC
0	LR-train	0.888002	0.910527	0.959692	0.746326
1	LR-test	0.881404	0.903697	0.959767	0.726278
2	DecisionTree-train	0.998476	0.999192	0.998976	0.997486
3	DecisionTree-test	0.832594	0.898838	0.900145	0.698871
4	KNN(7)-train	0.904726	0.905889	0.988145	0.739869
5	KNN(7)-test	0.879790	0.886553	0.981095	0.679250
6	NB-train	0.840836	0.939341	0.864526	0.794019
7	NB-test	0.847923	0.943218	0.869607	0.804996

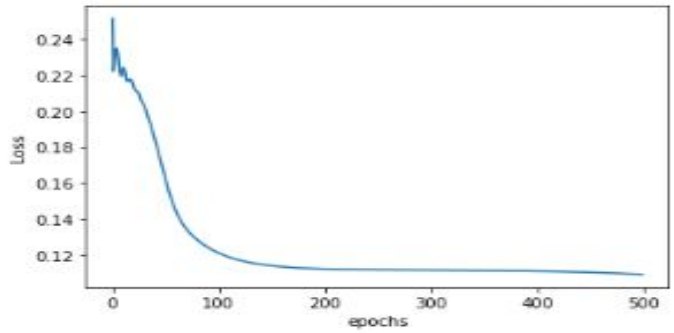
- Linear Models like KNN , NB and LR perform very well on GloVe embeddings
- LR being the best model since it has the low bias and low variance characteristic.
- Non-Linear Algos like Decision Tree tend to Overfit

GloVe with multilayer FeedForward Neural Network

Network Configuration - I: [50,100,100,1] (Neurons in each Layer)

LOSS CURVE

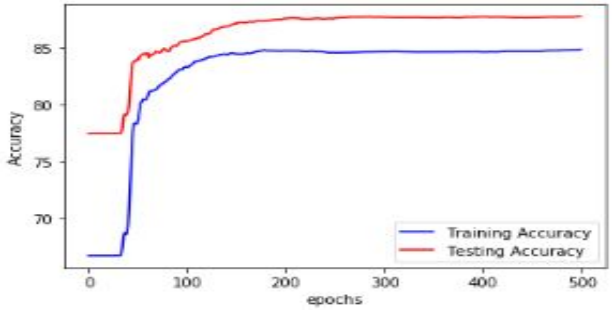
Network: [50,100,100,1] , Loss fn: MSE , Optimizer: Adam , LR: 0.001 , Epochs: 500 , Batch_size= 9000



This depicts the rate of change of loss with each epoch for the given model

ACCURACY VS EPOCH

Network: [50,100,100,1] , Loss fn: MSE , Optimizer: Adam , LR: 0.001 , Epochs: 500 , Batch_size= 9000



This depicts the change of Train and Test set accuracy with each epoch for the given model

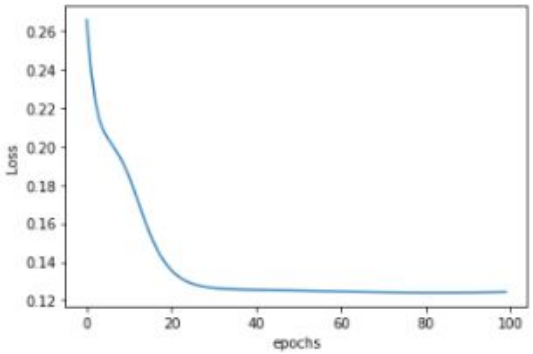
TRAIN - TEST ACCURACY

```
In [69]: #Train Accuracy
         ((fn(X_train_tensor.float())>0.5).type(torch.int)==y_train_tensor).sum().item()/y_train.shape[0]*100
Out[69]: 84.86666666666667

In [70]: #Test Accuracy
         ((fn(X_test_tensor.float())>0.5).type(torch.int)==y_test_tensor).sum().item()/y_test.shape[0]*100
Out[70]: 87.77842339724967
```

Network Configuration - II: [50,16,8,1] (Neurons in each Layer)

LOSS CURVE

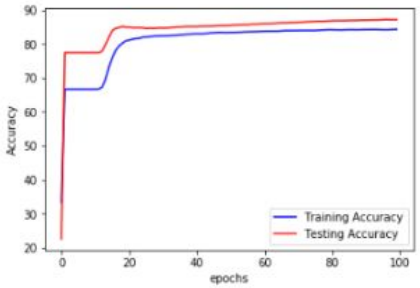
Network: [50,16,8,1] , Loss fn: MSE , Optimizer: Adam , LR: 0.0001 , Epochs: 100 , Batch_size= 30



This depicts the rate of change of loss with each epoch for the given model

ACCURACY VS EPOCH

Network: [50,16,8,1] , Loss fn: MSE , Optimizer: Adam , LR: 0.0001 , Epochs: 100 , Batch_size= 30



TRAIN-TEST ACCURACY (FINAL)

```
In [23]: #Train Accuracy
         ((fn(X_train_tensor.float())>0.5).type(torch.int)==y_train_tensor).sum().item()/y_train.shape[0]*100
Out[23]: 84.86666666666667

In [24]: #Test Accuracy
         ((fn(X_test_tensor.float())>0.5).type(torch.int)==y_test_tensor).sum().item()/y_test.shape[0]*100
Out[24]: 87.83652914971915
```

Performance on out of vocabulary words

```
"you are my inspiration sir" -  Clean
"die bitch, die" -  Hate
"you are so smart lol" -  Hate
"all jews are swine" -  Clean
"you are a friendly person" -  Clean
"shut up nigger" -  Hate
"black people deserve to burn in hell" -  Hate
"what are you looking at , dumb fuck !" -  Hate
"good people are found everywhere" -  Clean
```

- GloVe embeddings along with Neural Networks with the above Hyperparameters gives us the best result on both the Test as well as Out Of Vocab data as well.
- In the GloVe model we are able to capture the semantic meanings of individual words which helps us to achieve this result.
- The accuracy of Test being higher than Train only indicates the data imbalance issue.

Conclusion

- The project done by us aims to detect hate speech using a Natural Language Processing Technique.
- The dataset contains tweet annotated in two labels: hate/offensive and neither. The dataset was highly imbalanced with 70.6% hate tweets. The dataset was divided into train set and test set such that train set had 9000 tweets(66.67% hate, 33.33% non-hate) and test set had 5163 tweets(77.47% hate, 22.53% non-hate).
- Two models were used to detect hate speech:
  - Model 1 uses feature extraction using NLP techniques and applying linear models. N-grams and TF-IDF is used for vectorisation of tweets. The best result of this model was achieved by applying Logistic Regression which yields train set accuracy:89.83% and test set accuracy:89.07%.
  - Model 2 uses fully connected feed forward neural network. GloVe is used for vector representation of each tweet. We have created 50 dimension vector for each tweet. Fully connected neural network of [50,16,8,1] neurons in each layer gives train set accuracy of 84.86% and test set accuracy of 87.83%.

Future Work

- With such a highly imbalanced dataset we have yield a pretty good result. So a dataset richer in both quality and size would yield better performance.
- TFIDF cannot handle words out of vocabulary. Also, it generates a sparse vector. Sparse data requires better algorithms and optimization techniques for dimension reduction.
- GloVe is not dynamic. Also if a word is not present in the corpus of GloVe, it will fail to generate the vector of that word. We can use BERT in place of GloVe. Though BERT preserves the semantic context of a sentence, the sense of a word is lost.
- Multi Layered Perceptron (MLP) only preserves the context of a sentence. Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN) can be used in place of MLP as it preserves the order of the words and also provides better preservation of the context meaning.

Thank You!