Indian Institute of Information Technology Dharwad
Department of Computer Science Engineering

Mini Project Report

Multi-Label Hate Speech Detection

Done By :
K.V. S Bharadwaj (19BCS047)
P. Satwik (19BCS083)
P. Rutwik (19BCS084)
R. Dhatri Kiran Reddy (19BCS093)

Under the Supervision of :
Prof. Dr. Sunil Saumya (Ph.D CSE, NIT Patna)

Abstract

The correlation between opinions and text has been an enhancing topic for centuries. Today the web has become an excellent means of expressing opinions about anything, especially with the increasing popularity of social media. Such social media platform seems to be a proper platform for hate speech and hateful comments. Hate Speech is commonly defined as any form of communication that represents an individual or a group based on some characteristics such as race, color, ethnicity, gender etc. Any form of speech that hurts feelings of others, it might be an individual or a group. Due to the massive size of online contents that is being produced on a daily basis. As automated computational approaches based on machine learning techniques became a popular means of tackling these hateful speeches, feature selection plays an important role in terms of performance of the classification. In this report, we intend to explore the different combinations of features, including state-of-the-art models such as BERT and Bi-LSTM.

Introduction

Hate speech is a form of offending public speech targeted at particular individuals or groups of people on the basis of characteristics, such as race, directed vs generalized, religion, national origin, violence, disability, sexual orientation, or gender identity. In our day-to-day life, especially in social media, the hate speech is often accompanied with abusive language. Enough Research have been put on the investigation of algorithmic resolution of this complication. This leads to a state of confusion within the research community. The inconsistency can be considered as an inhibitor for the domain. To remedy this effect, this report introduces a novel configurable, multi-view approach to detect abusive language. We will show the process of preprocessing the multi-label dataset along with its results from various models.

Motivation

Social media has seen immense growth since the past decade, in its importance as a form of communication. The main vow of social media is that any individual can post whatever they want in any format, whether it is good, bad or anywhere between. Depending on the convention, such posts can be seen by millions of people. Different conventions have different characterization of inappropriate content and different methods for identifying it, but the scale of the medium means that automated methods are an important part of this task.

Hate-speech is a complex term with no single definition. Irrespective of the definition of the term, it is clear that automated methods for detecting hate-speech are necessary in some circumstances. In such cases it is critical that the methods applied are accurate, effective, and efficient.

The main objectives of this report was :

- To explore different combinations of features with classical ML techniques for hate speech detection from text.
- To check the performance of such classifiers using embeddings from BERT/GloVe/Word2Vec
- To undertake an additional investigation to understand the impact of external data and feature engineering in classic methods in the domain of multilingual hate speech.

Related Works

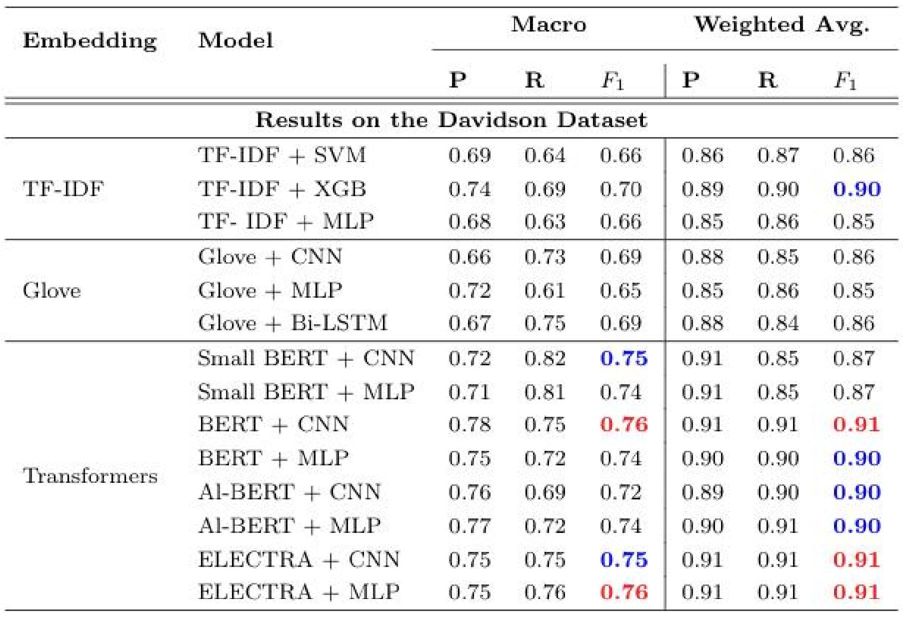
The literature reviews in this study are arranged in a way that returns both feature variations and modelling techniques.

1) **Word Embedding Techniques** -

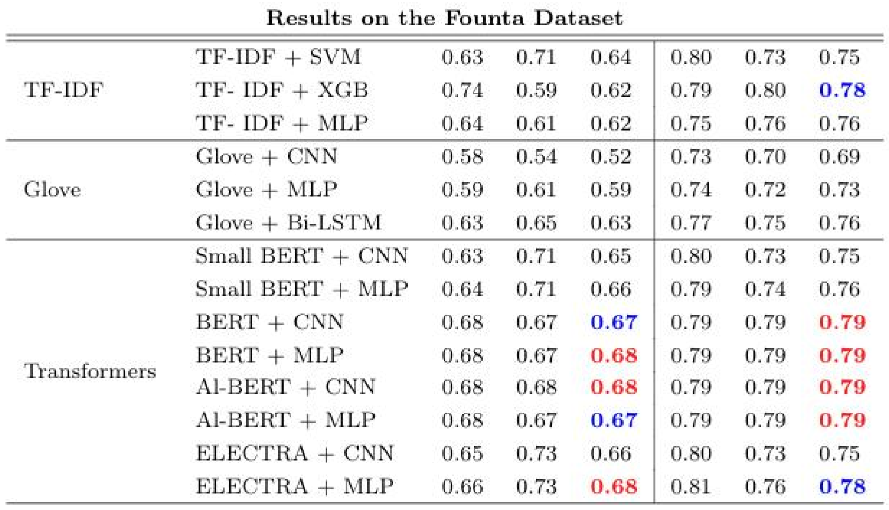
The most widespread embedding method employed in natural language processing is the word embedding. It can be seen as a two-layer Neural Network, trained to reconstruct linguistic contexts of words.This in terms produces a more cultivated version of a vector where semantically related words may end up having similar vector representations relative to each other. The word vectors can then be used as a classification feature and have been shown to outperform bag-of-words approaches for a number of general NLP learning tasks as well as hate speech specific classification tasks. In this report we will see about the BERT model and GloVe.

2) **Machine Learning Techniques in Recent Shared Tasks** -

Three widely-used datasets, including Davidson, Founta and Twitter Sentiment Analysis, are adopted in previous experiments, with their key statistics shown in figure below. The macro and weighted average results of all 14 hate speech detection models on three popular benchmarks are shown below. ELECTRA shows superior performance across the three datasets; it is interesting that XGB using TF-IDF also obtains impressive performance.

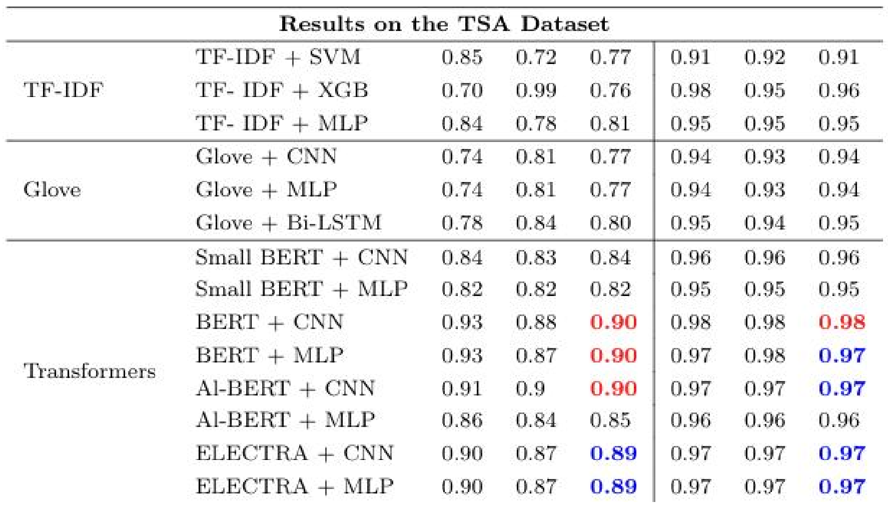


Embedding	Model	Macro			Weighted Avg.		
		P	R	F_1	P	R	F_1
	Results on the Davidson Dataset						
TF-IDF	TF-IDF + SVM	0.69	0.64	0.66	0.86	0.87	0.86
	TF-IDF + XGB	0.74	0.69	0.70	0.89	0.90	**0.90**
	TF- IDF + MLP	0.68	0.63	0.66	0.85	0.86	0.85
Glove	Glove + CNN	0.66	0.73	0.69	0.88	0.85	0.86
	Glove + MLP	0.72	0.61	0.65	0.85	0.86	0.85
	Glove + Bi-LSTM	0.67	0.75	0.69	0.88	0.84	0.86
Transformers	Small BERT + CNN	0.72	0.82	**0.75**	0.91	0.85	0.87
	Small BERT + MLP	0.71	0.81	0.74	0.91	0.85	0.87
	BERT + CNN	0.78	0.75	**0.76**	0.91	0.91	**0.91**
	BERT + MLP	0.75	0.72	0.74	0.90	0.90	**0.90**
	Al-BERT + CNN	0.76	0.69	0.72	0.89	0.90	**0.90**
	Al-BERT + MLP	0.77	0.72	0.74	0.90	0.91	**0.90**
	ELECTRA + CNN	0.75	0.75	**0.75**	0.91	0.91	**0.91**
	ELECTRA + MLP	0.75	0.76	**0.76**	0.91	0.91	**0.91**



Results on the Founta Dataset

TF-IDF	TF-IDF + SVM	0.63	0.71	0.64	0.80	0.73	0.75
	TF- IDF + XGB	0.74	0.59	0.62	0.79	0.80	**0.78**
	TF- IDF + MLP	0.64	0.61	0.62	0.75	0.76	0.76
Glove	Glove + CNN	0.58	0.54	0.52	0.73	0.70	0.69
	Glove + MLP	0.59	0.61	0.59	0.74	0.72	0.73
	Glove + Bi-LSTM	0.63	0.65	0.63	0.77	0.75	0.76
Transformers	Small BERT + CNN	0.63	0.71	0.65	0.80	0.73	0.75
	Small BERT + MLP	0.64	0.71	0.66	0.79	0.74	0.76
	BERT + CNN	0.68	0.67	**0.67**	0.79	0.79	**0.79**
	BERT + MLP	0.68	0.67	**0.68**	0.79	0.79	**0.79**
	Al-BERT + CNN	0.68	0.68	**0.68**	0.79	0.79	**0.79**
	Al-BERT + MLP	0.68	0.67	**0.67**	0.79	0.79	**0.79**
	ELECTRA + CNN	0.65	0.73	0.66	0.80	0.73	0.75
	ELECTRA + MLP	0.66	0.73	**0.68**	0.81	0.76	**0.78**



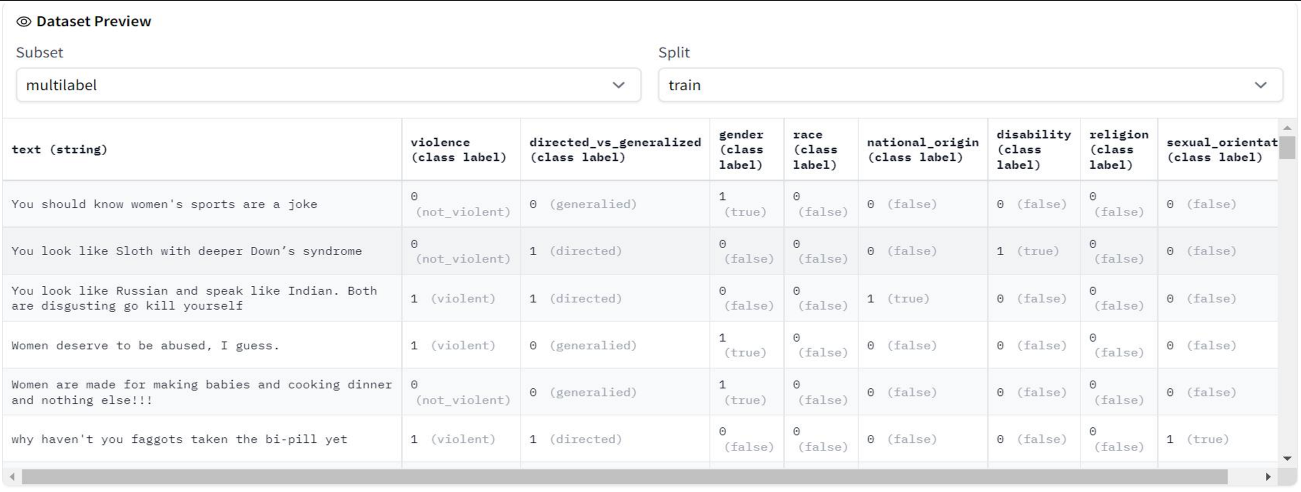
Results on the TSA Dataset

TF-IDF	TF-IDF + SVM	0.85	0.72	0.77	0.91	0.92	0.91
	TF- IDF + XGB	0.70	0.99	0.76	0.98	0.95	0.96
	TF- IDF + MLP	0.84	0.78	0.81	0.95	0.95	0.95
Glove	Glove + CNN	0.74	0.81	0.77	0.94	0.93	0.94
	Glove + MLP	0.74	0.81	0.77	0.94	0.93	0.94
	Glove + Bi-LSTM	0.78	0.84	0.80	0.95	0.94	0.95
Transformers	Small BERT + CNN	0.84	0.83	0.84	0.96	0.96	0.96
	Small BERT + MLP	0.82	0.82	0.82	0.95	0.95	0.95
	BERT + CNN	0.93	0.88	**0.90**	0.98	0.98	**0.98**
	BERT + MLP	0.93	0.87	**0.90**	0.97	0.98	**0.97**
	Al-BERT + CNN	0.91	0.9	**0.90**	0.97	0.97	**0.97**
	Al-BERT + MLP	0.86	0.84	0.85	0.96	0.96	0.96
	ELECTRA + CNN	0.90	0.87	**0.89**	0.97	0.97	**0.97**
	ELECTRA + MLP	0.90	0.87	**0.89**	0.97	0.97	**0.97**

Dataset Description

Hate speech is a highly suave phenomenon which is heavily dependent on the individuals background and the data being analyzed. Moreover, there is no clear standardization when it comes to the sub-categorization of hateful speech, and almost all the available datasets differ in the way they have looked into the problem and categorized them. Thus each of the datasets has its own version of class labels, and it is difficult to use external data that really helps. Nevertheless, we have used ETHOS Dataset and Jigsaw Toxic Comments Kaggle dataset.

1) **ETHOS** - It is a hate speech detection dataset. It is made from YouTube and Reddit comments authenticated through a collaborating platform. It has two subsets, one for binary classification and the other for multi-label classification. The binary classification contains 998 comments, while the multi-label contains fine-grained hate-speech annotations for 433 comments. We will be using the multi-label dataset with 433 comments for our project. This dataset has around 8 labels and 433 sentences.



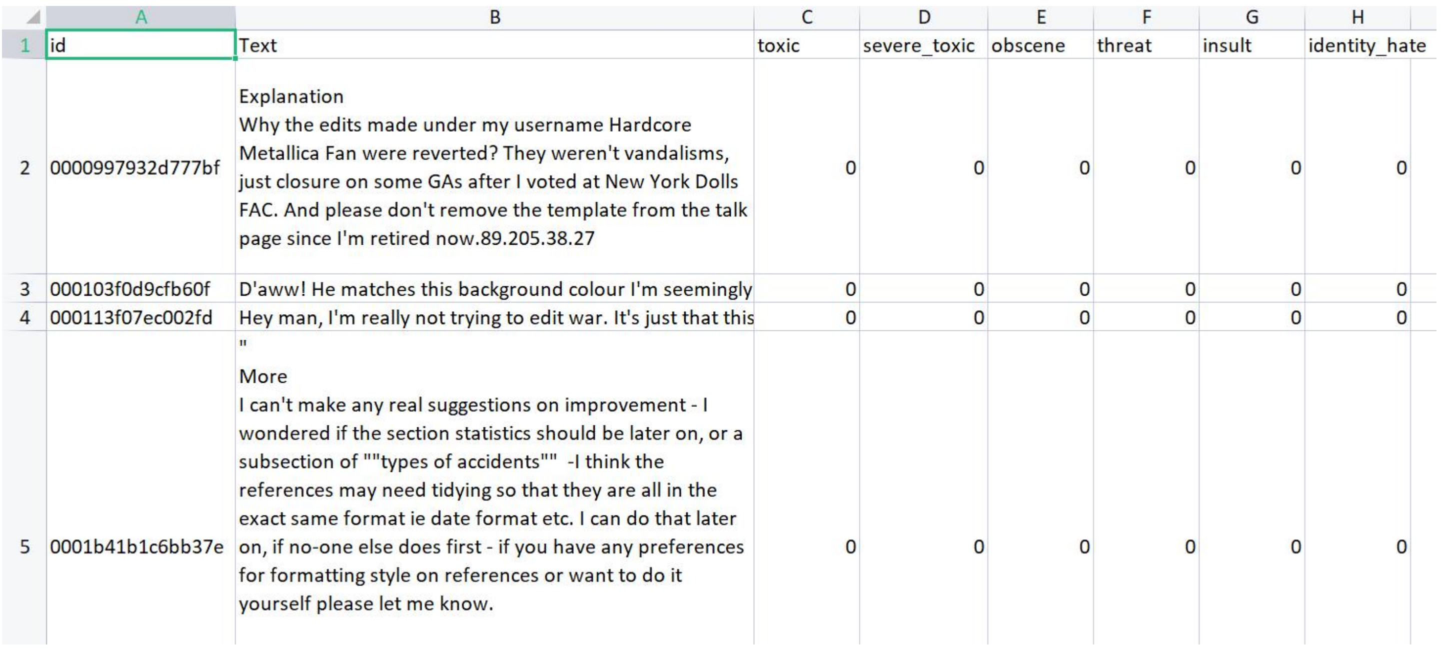
Dataset Preview

Subset: multilabel

Split: train

text (string)	violence (class label)	directed_vs_generalized (class label)	gender (class label)	race (class label)	national_origin (class label)	disability (class label)	religion (class label)	sexual_orientat (class label)
You should know women's sports are a joke	0 (not_violent)	0 (generalied)	1 (true)	0 (false)	0 (false)	0 (false)	0 (false)	0 (false)
You look like Sloth with deeper Down's syndrome	0 (not_violent)	1 (directed)	0 (false)	0 (false)	0 (false)	1 (true)	0 (false)	0 (false)
You look like Russian and speak like Indian. Both are disgusting go kill yourself	1 (violent)	1 (directed)	0 (false)	0 (false)	1 (true)	0 (false)	0 (false)	0 (false)
Women deserve to be abused, I guess.	1 (violent)	0 (generalied)	1 (true)	0 (false)	0 (false)	0 (false)	0 (false)	0 (false)
Women are made for making babies and cooking dinner and nothing else!!!	0 (not_violent)	0 (generalied)	1 (true)	0 (false)	0 (false)	0 (false)	0 (false)	0 (false)
why haven't you faggots taken the bi-pill yet	1 (violent)	1 (directed)	0 (false)	0 (false)	0 (false)	0 (false)	0 (false)	1 (true)

2) Jigsaw Toxic Comment Classification - This dataset is also a multi-label dataset which is taken from a Kaggle competition. This dataset was contributed by Google. This dataset is a large collection of Wikipedia comments which have been manually labeled for toxicity. The hate speech is labeled into 6 classes and this dataset has around 160000 sentences.



	A	B	C	D	E	F	G	H
1	id	Text	toxic	severe_toxic	obscene	threat	insult	identity_hate
2	0000997932d777bf	Explanation Why the edits made under my username Hardcore Metallica Fan were reverted? They weren't vandalisms, just closure on some GAs after I voted at New York Dolls FAC. And please don't remove the template from the talk page since I'm retired now.89.205.38.27	0	0	0	0	0	0
3	000103f0d9cfb60f	D'aww! He matches this background colour I'm seemingly	0	0	0	0	0	0
4	000113f07ec002fd	Hey man, I'm really not trying to edit war. It's just that this	0	0	0	0	0	0
5	0001b41b1c6bb37e	" More I can't make any real suggestions on improvement - I wondered if the section statistics should be later on, or a subsection of ""types of accidents"" -I think the references may need tidying so that they are all in the exact same format ie date format etc. I can do that later on, if no-one else does first - if you have any preferences for formatting style on references or want to do it yourself please let me know.	0	0	0	0	0	0

Research Methodology

This subtopic will outline the technical detail and the methods used in the implementation of our system. At first, we will introduce the flowchart used for our methodology. Then we will talk about the pre-processing steps like Expanding the contractions, Lowercasing the text, Removing Punctuations, Digits, Stopwords etc. and Lemmatization. Then we will see about word vectorization using BERT and keras tokenizer. Then finally about our model.

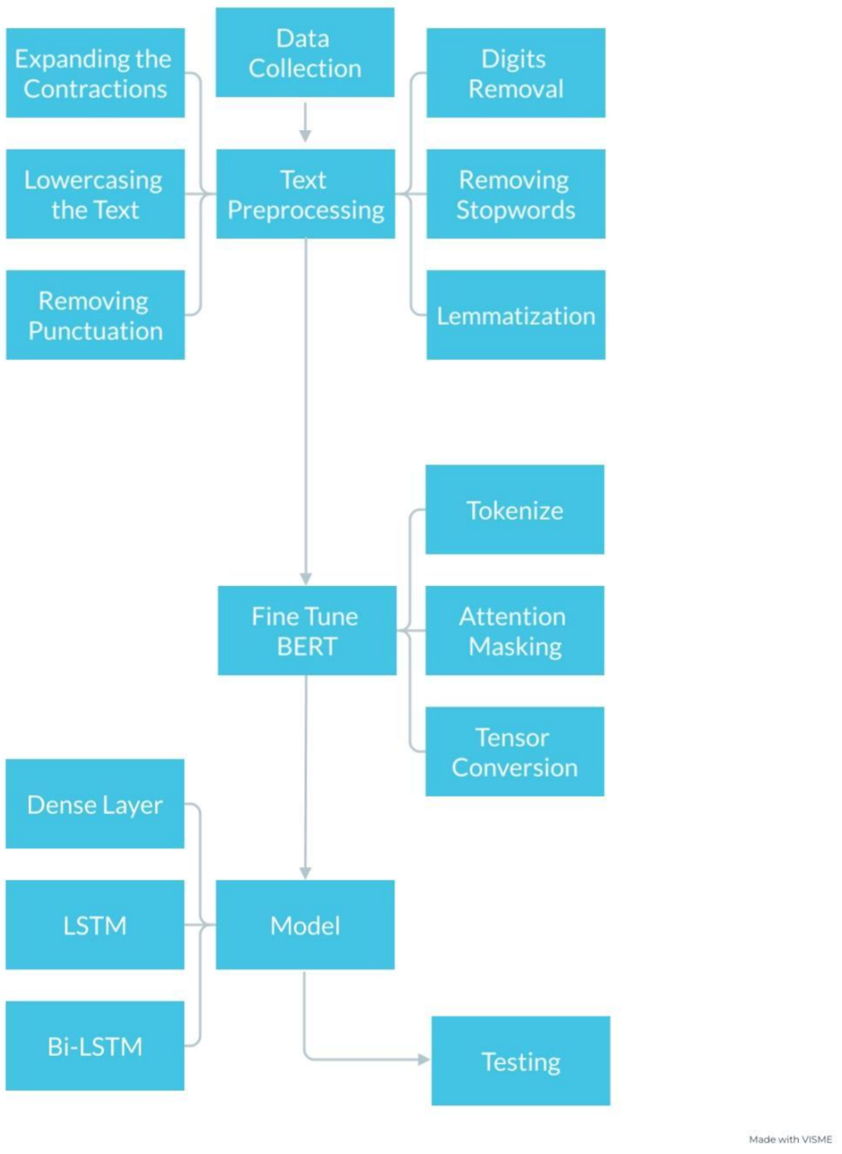


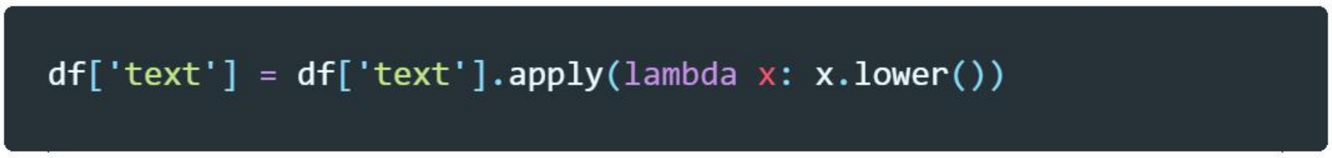
Fig: Methodology Flowchart

<u>Text PreProcessing</u>

1) Lowercasing the Text -

Converting the data to lowercase helps us in the process of preprocessing in the later stages of the NLP application. It is also easy for the machine to interpret the words because the lowercase and uppercase are treated differently by the machine because of their Ascii values.

Code Snippet :



```
df['text'] = df['text'].apply(lambda x: x.lower())
```

Example :

Input :

You look like **R**ussian and speak **L**ike **I**ndian.
Women deserved to be abused, **I** guess.

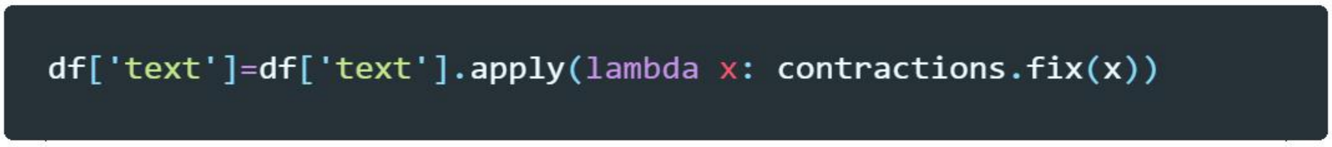
Output :

you look like **r**ussian and speak **l**ike **i**ndian.
women deserved to be abused, **i** guess.

2) Expanding the Contractions -

Contraction is the shortened form of a word. These occur by dropping letters or replacing them by an apostrophe(‘). Like this we need to expand this contraction in the text data for better analysis.

Code Snippet :



```
df['text']=df['text'].apply(lambda x: contractions.fix(x))
```

Example :

Input :

Why **haven’t** you faggots taken the bill yet
Why **don’t** you go beck your god to murder you ?

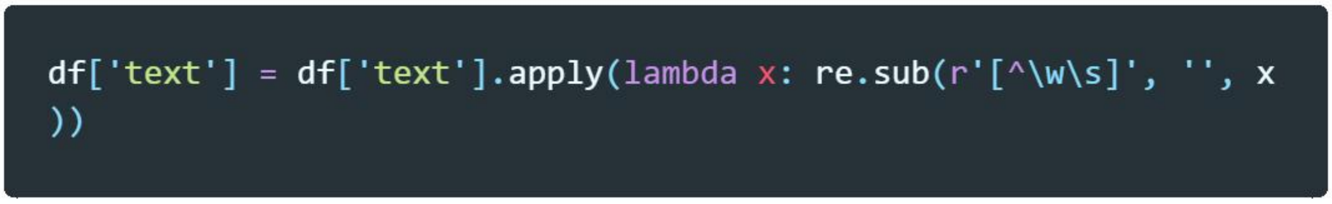
Output :

Why **have not** you faggots taken the bill yet
Why **do not** you go beck your god to murder you ?

3) Removing Punctuations -

Punctuations may affect result of any text processing approach, especially what depends on the occurrence of the frequencies of words and phrases,since the punctuation marks are used frequently in the text.

Code Snippet :



```
df['text'] = df['text'].apply(lambda x: re.sub(r'[^\w\s]', '', x
))
```

Example :

Input :

women are made for making babies and cooking dinner and nothing else**!!!**
Why does Bulgaria exist as a country**?**

Output :

women are made for making babies and cooking dinner and nothing else
Why does Bulgaria exist as a country

4) Removing the Digits -

This type of words or numbers are difficult to process, so it is better to remove them or replace them or replace them with an empty string. We use regular expressions for doing this.

Code Snippet :

```
df['text'] = df['text'].apply(lambda x: ''.join([i for i in x if
 not i.isdigit()]))
```

Example :

Input :

This sick disgusting white terrorist fascist killed more than **50** innocent people

Output :

This sick disgusting white terrorist fascist killed more than innocent people

5) Removing StopWords -

Stopwords are most commonly occurring words in a text which do not provide any valuable information. Stopwords like they, there, this, where etc are some of the stop-words. There is a library named NLTK that can be used to remove stopwords and includes approximately 180 stopwords which it removes.

Code Snippet :

```
nltk.download("stopwords")
from nltk.corpus import stopwords
stop_words = set(stopwords.words('english'))
def remove_stopwords(text):
    return " ".join([word for word in str(text).split() if word
not in stop_words])
df['text'] = df['text'].apply(lambda x: remove_stopwords(x))
```

Example :

Input :

you should know womens sports **are a** joke

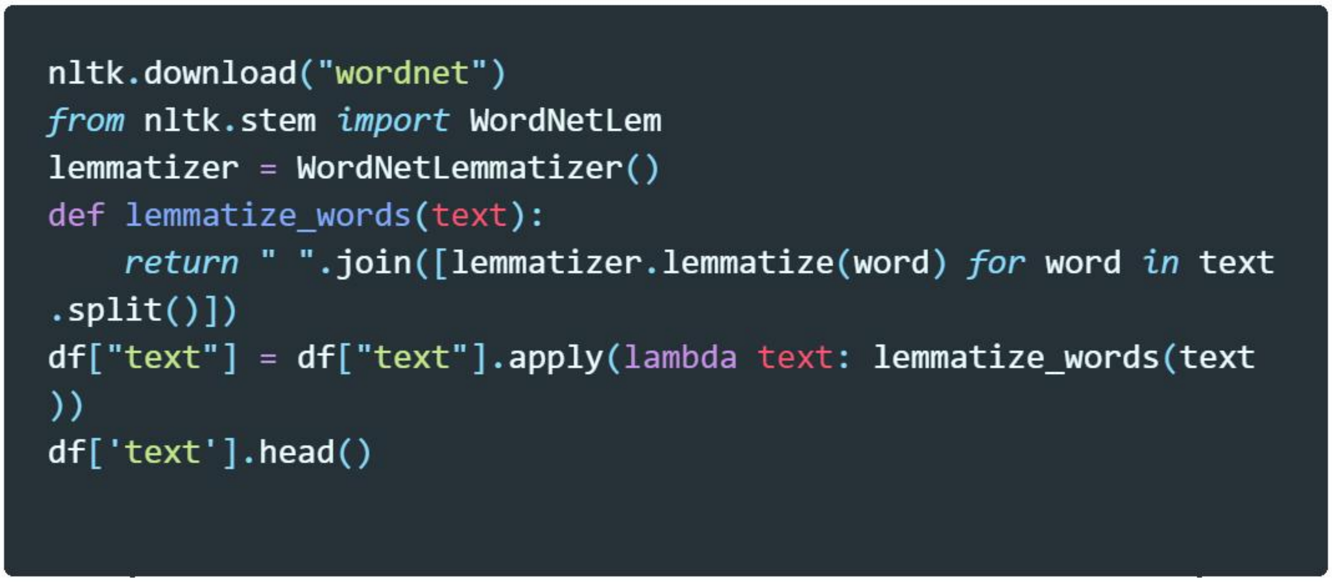
Output :

know womens sports joke

6) Lemmatization -

Lemmatization usually refers to doing things properly with the use of a vocabulary and linguistic segmentation of words, normally aiming to remove conjugational endings only and to return the dictionary form of a word, which is known as the lemma .

Code Snippet :



```
nltk.download("wordnet")
from nltk.stem import WordNetLem
lemmatizer = WordNetLemmatizer()
def lemmatize_words(text):
    return " ".join([lemmatizer.lemmatize(word) for word in text
.split()])
df["text"] = df["text"].apply(lambda text: lemmatize_words(text
))
df['text'].head()
```

Example :

Input :

know **womens** sports joke

look**s** like sloth deeper down syndrome

Output :

know **woman** sport joke

look like sloth deeper down syndrome

Models

1) Bi - LSTM **-**

Bidirectional recurrent neural networks(RNN) is nothing but putting two independent RNNs together. This structure allows the neural networks to have both backward and forward information throughout the sequence at every time step. Using bidirectional, it will run your inputs in two ways, one will be from past to future and other from future to past. What differs this approach from unidirectional is that in the LSTM that runs backward you preserve information from the future and using the two hidden states combined you are able in any point in time to preserve information from both past and future.

2) LSTM -

LSTM is a recurrent neural network (RNN) architecture that remembers values over capricious intervals. LSTM is well-suited to classify, process and predict time series given time lags of unknown duration. Prenominal to gap length gives an advantage to LSTM over alternative RNNs, hidden Markov models and other sequence learning methods.

3) Dense Layers -

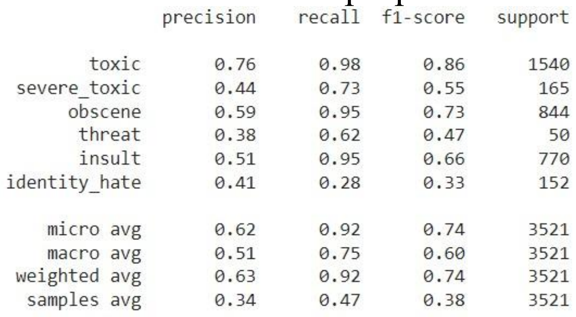
Dense layer is the deeply connected NN layer. It is the most common and frequently used layer.

4) ULMFit -

Universal Language Model Fine-Tuning(ULMFIT) is a transfer learning technique which can help in various NLP tasks. It has been state-of-the-art NLP technique for a long time.

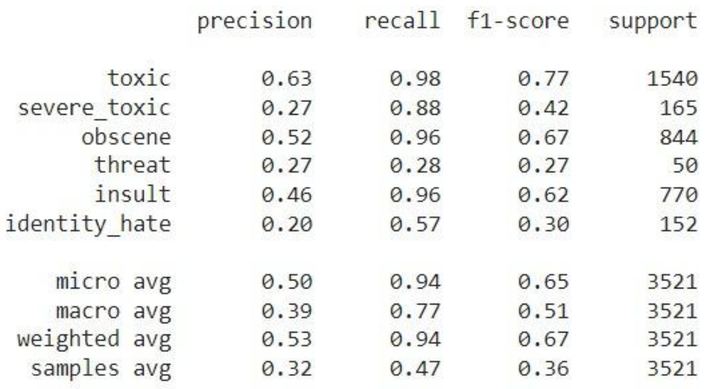
Results

The Results have been proper for the ULMFit Model.



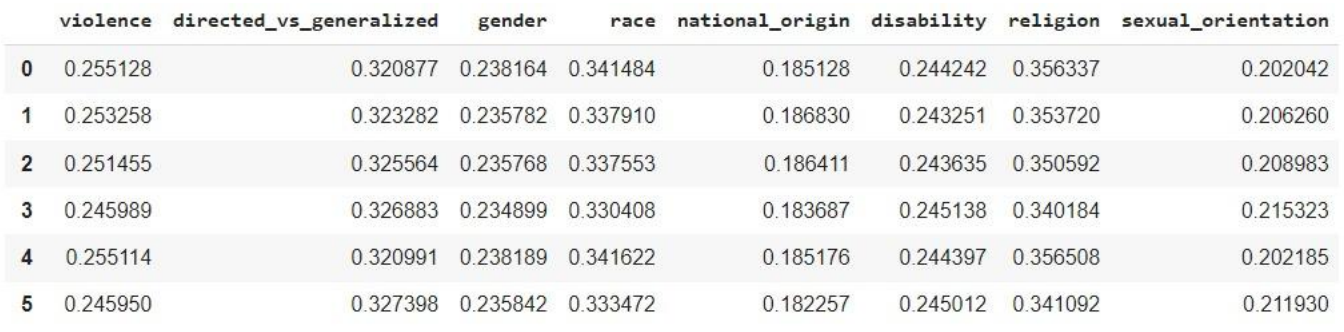
```
               precision    recall  f1-score   support

        toxic       0.76      0.98      0.86      1540
 severe_toxic       0.44      0.73      0.55       165
      obscene       0.59      0.95      0.73       844
       threat       0.38      0.62      0.47        50
       insult       0.51      0.95      0.66       770
identity_hate       0.41      0.28      0.33       152

    micro avg       0.62      0.92      0.74      3521
    macro avg       0.51      0.75      0.60      3521
 weighted avg       0.63      0.92      0.74      3521
  samples avg       0.34      0.47      0.38      3521
```

Fig : Result when preprocessing was not applied on Jigsaw Dataset



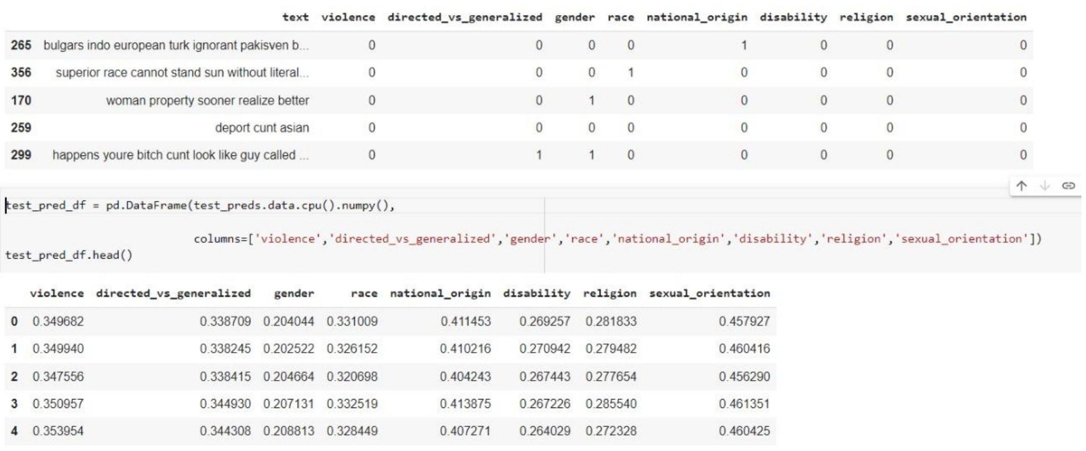
```
               precision    recall  f1-score   support

        toxic       0.63      0.98      0.77      1540
 severe_toxic       0.27      0.88      0.42       165
      obscene       0.52      0.96      0.67       844
       threat       0.27      0.28      0.27        50
       insult       0.46      0.96      0.62       770
identity_hate       0.20      0.57      0.30       152

    micro avg       0.50      0.94      0.65      3521
    macro avg       0.39      0.77      0.51      3521
 weighted avg       0.53      0.94      0.67      3521
  samples avg       0.32      0.47      0.36      3521
```

Fig : Result when preprocessing was applied on Jigsaw Dataset



	violence	directed_vs_generalized	gender	race	national_origin	disability	religion	sexual_orientation
0	0.255128	0.320877	0.238164	0.341484	0.185128	0.244242	0.356337	0.202042
1	0.253258	0.323282	0.235782	0.337910	0.186830	0.243251	0.353720	0.206260
2	0.251455	0.325564	0.235768	0.337553	0.186411	0.243635	0.350592	0.208983
3	0.245989	0.326883	0.234899	0.330408	0.183687	0.245138	0.340184	0.215323
4	0.255114	0.320991	0.238189	0.341622	0.185176	0.244397	0.356508	0.202185
5	0.245950	0.327398	0.235842	0.333472	0.182257	0.245012	0.341092	0.211930

Fig : Result when preprocessing was not applied on ethos Dataset



	text	violence	directed_vs_generalized	gender	race	national_origin	disability	religion	sexual_orientation
265	bulgars indo european turk ignorant pakisven b...	0	0	0	0	1	0	0	0
356	superior race cannot stand sun without literal...	0	0	0	1	0	0	0	0
170	woman property sooner realize better	0	0	1	0	0	0	0	0
259	deport cunt asian	0	0	0	0	0	0	0	0
299	happens youre bitch cunt look like guy called ...	0	1	1	0	0	0	0	0

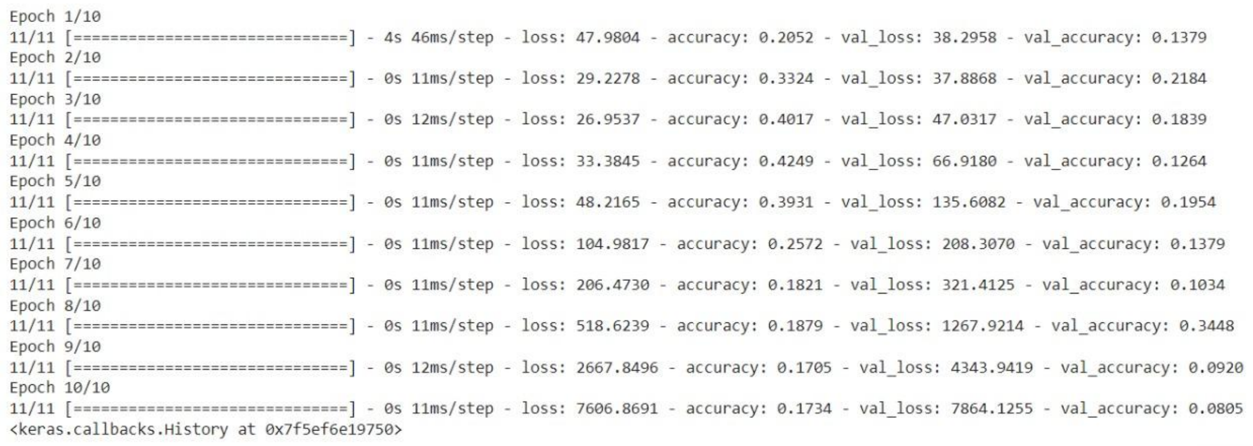
```python
test_pred_df = pd.DataFrame(test_preds.data.cpu().numpy(),

                         columns=['violence','directed_vs_generalized','gender','race','national_origin','disability','religion','sexual_orientation'])
test_pred_df.head()
```

	violence	directed_vs_generalized	gender	race	national_origin	disability	religion	sexual_orientation
0	0.349682	0.338709	0.204044	0.331009	0.411453	0.269257	0.281833	0.457927
1	0.349940	0.338245	0.202522	0.326152	0.410216	0.270942	0.279482	0.460416
2	0.347556	0.338415	0.204664	0.320698	0.404243	0.267443	0.277654	0.456290
3	0.350957	0.344930	0.207131	0.332519	0.413875	0.267226	0.285540	0.461351
4	0.353954	0.344308	0.208813	0.328449	0.407271	0.264029	0.272328	0.460425

Fig : Result when preprocessing was applied on ethos Dataset

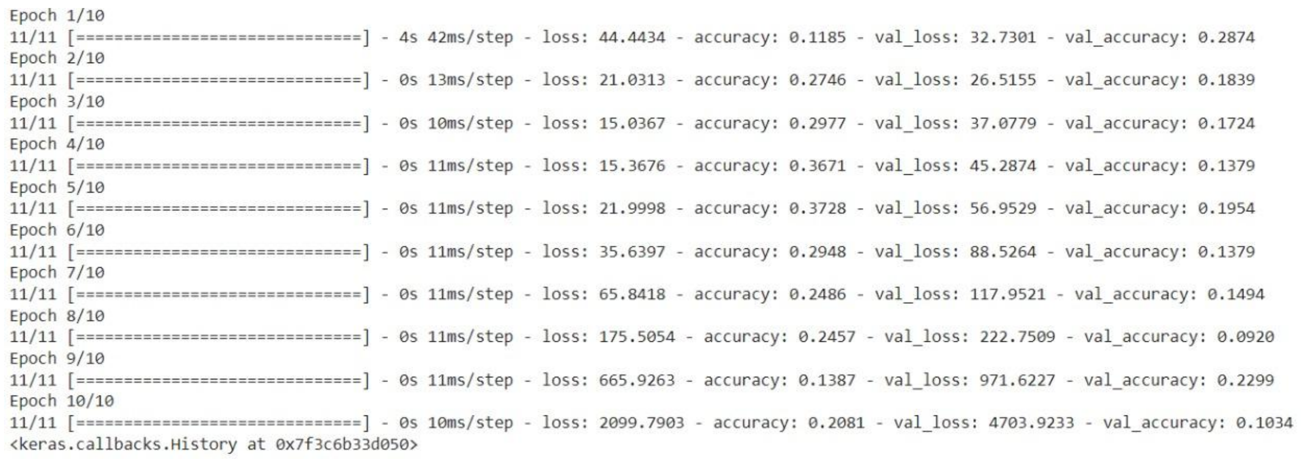
Challenges

As the size of the Ethos Dataset is very small (433 comments), We were unable to get proper results for certain models. The results which we got for various models are shown below :



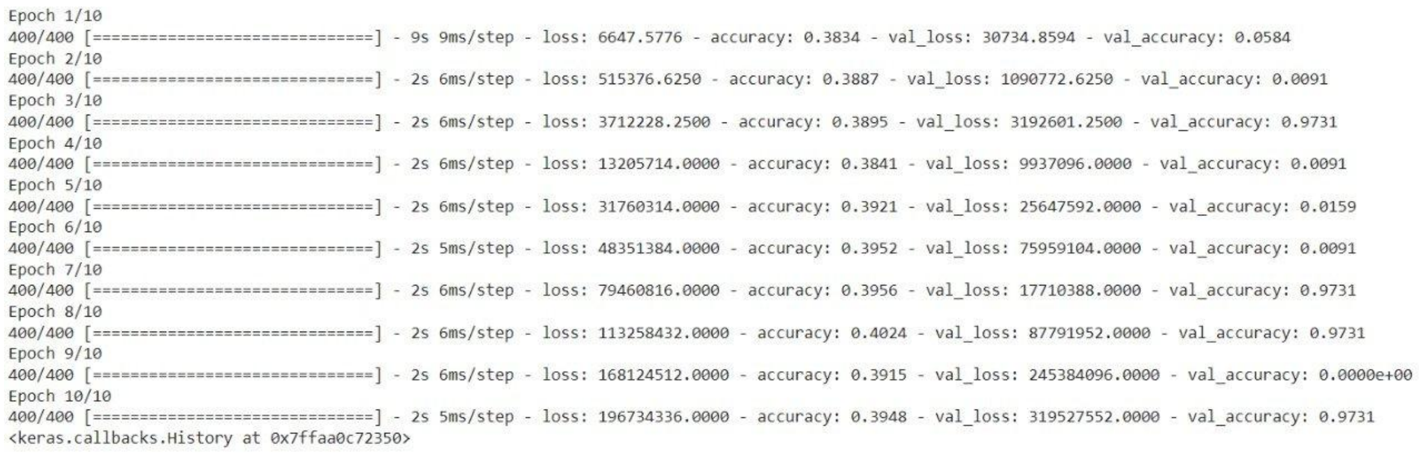
```
Epoch 1/10
11/11 [==============================] - 4s 46ms/step - loss: 47.9804 - accuracy: 0.2052 - val_loss: 38.2958 - val_accuracy: 0.1379
Epoch 2/10
11/11 [==============================] - 0s 11ms/step - loss: 29.2278 - accuracy: 0.3324 - val_loss: 37.8868 - val_accuracy: 0.2184
Epoch 3/10
11/11 [==============================] - 0s 12ms/step - loss: 26.9537 - accuracy: 0.4017 - val_loss: 47.0317 - val_accuracy: 0.1839
Epoch 4/10
11/11 [==============================] - 0s 11ms/step - loss: 33.3845 - accuracy: 0.4249 - val_loss: 66.9180 - val_accuracy: 0.1264
Epoch 5/10
11/11 [==============================] - 0s 11ms/step - loss: 48.2165 - accuracy: 0.3931 - val_loss: 135.6082 - val_accuracy: 0.1954
Epoch 6/10
11/11 [==============================] - 0s 11ms/step - loss: 104.9817 - accuracy: 0.2572 - val_loss: 208.3070 - val_accuracy: 0.1379
Epoch 7/10
11/11 [==============================] - 0s 11ms/step - loss: 206.4730 - accuracy: 0.1821 - val_loss: 321.4125 - val_accuracy: 0.1034
Epoch 8/10
11/11 [==============================] - 0s 11ms/step - loss: 518.6239 - accuracy: 0.1879 - val_loss: 1267.9214 - val_accuracy: 0.3448
Epoch 9/10
11/11 [==============================] - 0s 12ms/step - loss: 2667.8496 - accuracy: 0.1705 - val_loss: 4343.9419 - val_accuracy: 0.0920
Epoch 10/10
11/11 [==============================] - 0s 11ms/step - loss: 7606.8691 - accuracy: 0.1734 - val_loss: 7864.1255 - val_accuracy: 0.0805
<keras.callbacks.History at 0x7f5ef6e19750>
```

Fig : Result when keras tokenizer was used without preprocessing the data on ethos dataset



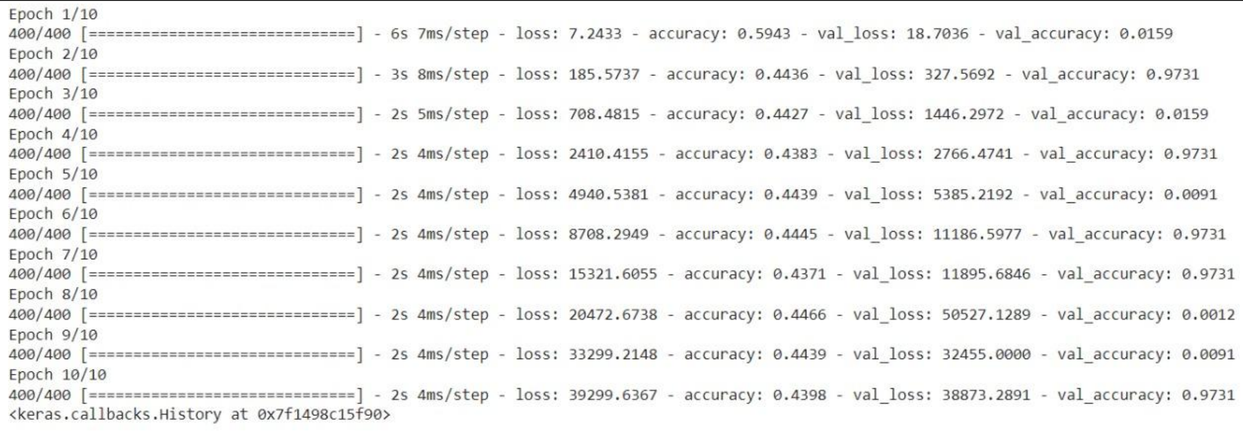
```
Epoch 1/10
11/11 [==============================] - 4s 42ms/step - loss: 44.4434 - accuracy: 0.1185 - val_loss: 32.7301 - val_accuracy: 0.2874
Epoch 2/10
11/11 [==============================] - 0s 13ms/step - loss: 21.0313 - accuracy: 0.2746 - val_loss: 26.5155 - val_accuracy: 0.1839
Epoch 3/10
11/11 [==============================] - 0s 10ms/step - loss: 15.0367 - accuracy: 0.2977 - val_loss: 37.0779 - val_accuracy: 0.1724
Epoch 4/10
11/11 [==============================] - 0s 11ms/step - loss: 15.3676 - accuracy: 0.3671 - val_loss: 45.2874 - val_accuracy: 0.1379
Epoch 5/10
11/11 [==============================] - 0s 11ms/step - loss: 21.9998 - accuracy: 0.3728 - val_loss: 56.9529 - val_accuracy: 0.1954
Epoch 6/10
11/11 [==============================] - 0s 11ms/step - loss: 35.6397 - accuracy: 0.2948 - val_loss: 88.5264 - val_accuracy: 0.1379
Epoch 7/10
11/11 [==============================] - 0s 11ms/step - loss: 65.8418 - accuracy: 0.2486 - val_loss: 117.9521 - val_accuracy: 0.1494
Epoch 8/10
11/11 [==============================] - 0s 11ms/step - loss: 175.5054 - accuracy: 0.2457 - val_loss: 222.7509 - val_accuracy: 0.0920
Epoch 9/10
11/11 [==============================] - 0s 11ms/step - loss: 665.9263 - accuracy: 0.1387 - val_loss: 971.6227 - val_accuracy: 0.2299
Epoch 10/10
11/11 [==============================] - 0s 10ms/step - loss: 2099.7903 - accuracy: 0.2081 - val_loss: 4703.9233 - val_accuracy: 0.1034
<keras.callbacks.History at 0x7f3c6b33d050>
```

Fig : Result when keras tokenizer was used with preprocessing the data on ethos dataset



```
Epoch 1/10
400/400 [==============================] - 9s 9ms/step - loss: 6647.5776 - accuracy: 0.3834 - val_loss: 30734.8594 - val_accuracy: 0.0584
Epoch 2/10
400/400 [==============================] - 2s 6ms/step - loss: 515376.6250 - accuracy: 0.3887 - val_loss: 1090772.6250 - val_accuracy: 0.0091
Epoch 3/10
400/400 [==============================] - 2s 6ms/step - loss: 3712228.2500 - accuracy: 0.3895 - val_loss: 3192601.2500 - val_accuracy: 0.9731
Epoch 4/10
400/400 [==============================] - 2s 6ms/step - loss: 13205714.0000 - accuracy: 0.3841 - val_loss: 9937096.0000 - val_accuracy: 0.0091
Epoch 5/10
400/400 [==============================] - 2s 6ms/step - loss: 31760314.0000 - accuracy: 0.3921 - val_loss: 25647592.0000 - val_accuracy: 0.0159
Epoch 6/10
400/400 [==============================] - 2s 5ms/step - loss: 48351384.0000 - accuracy: 0.3952 - val_loss: 75959104.0000 - val_accuracy: 0.0091
Epoch 7/10
400/400 [==============================] - 2s 6ms/step - loss: 79460816.0000 - accuracy: 0.3956 - val_loss: 17710388.0000 - val_accuracy: 0.9731
Epoch 8/10
400/400 [==============================] - 2s 6ms/step - loss: 113258432.0000 - accuracy: 0.4024 - val_loss: 87791952.0000 - val_accuracy: 0.9731
Epoch 9/10
400/400 [==============================] - 2s 6ms/step - loss: 168124512.0000 - accuracy: 0.3915 - val_loss: 245384096.0000 - val_accuracy: 0.0000e+00
Epoch 10/10
400/400 [==============================] - 2s 5ms/step - loss: 196734336.0000 - accuracy: 0.3948 - val_loss: 319527552.0000 - val_accuracy: 0.9731
<keras.callbacks.History at 0x7ffaa0c72350>
```

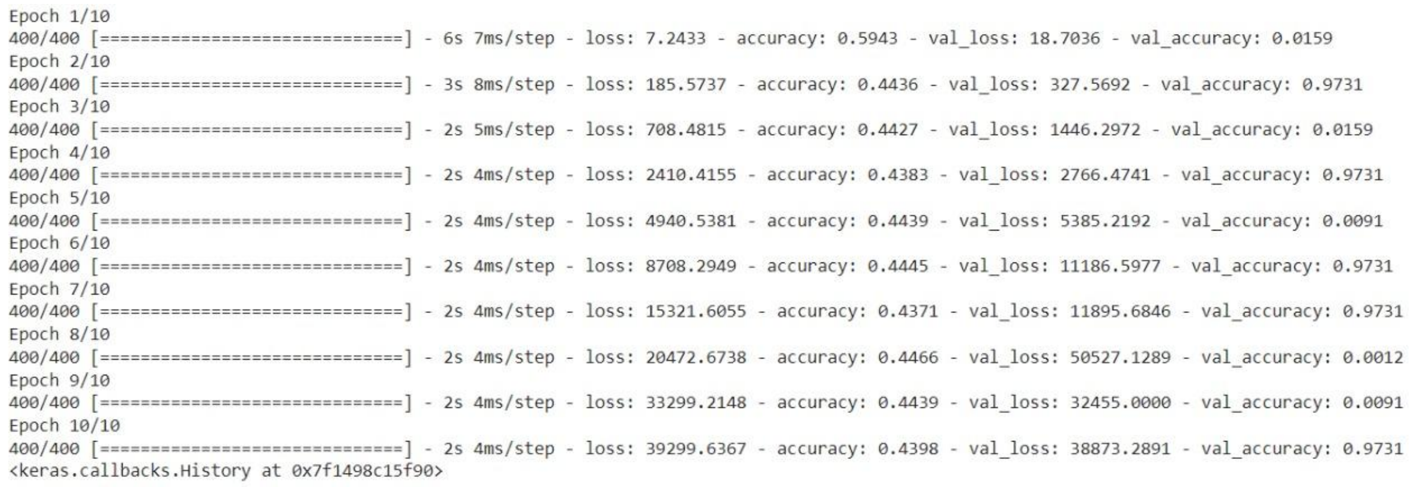
Fig : Result when keras tokenizer was used without preprocessing the data on jigsaw dataset



```
Epoch 1/10
400/400 [==============================] - 6s 7ms/step - loss: 7.2433 - accuracy: 0.5943 - val_loss: 18.7036 - val_accuracy: 0.0159
Epoch 2/10
400/400 [==============================] - 3s 8ms/step - loss: 185.5737 - accuracy: 0.4436 - val_loss: 327.5692 - val_accuracy: 0.9731
Epoch 3/10
400/400 [==============================] - 2s 5ms/step - loss: 708.4815 - accuracy: 0.4427 - val_loss: 1446.2972 - val_accuracy: 0.0159
Epoch 4/10
400/400 [==============================] - 2s 4ms/step - loss: 2410.4155 - accuracy: 0.4383 - val_loss: 2766.4741 - val_accuracy: 0.9731
Epoch 5/10
400/400 [==============================] - 2s 4ms/step - loss: 4940.5381 - accuracy: 0.4439 - val_loss: 5385.2192 - val_accuracy: 0.0091
Epoch 6/10
400/400 [==============================] - 2s 4ms/step - loss: 8708.2949 - accuracy: 0.4445 - val_loss: 11186.5977 - val_accuracy: 0.9731
Epoch 7/10
400/400 [==============================] - 2s 4ms/step - loss: 15321.6055 - accuracy: 0.4371 - val_loss: 11895.6846 - val_accuracy: 0.9731
Epoch 8/10
400/400 [==============================] - 2s 4ms/step - loss: 20472.6738 - accuracy: 0.4466 - val_loss: 50527.1289 - val_accuracy: 0.0012
Epoch 9/10
400/400 [==============================] - 2s 4ms/step - loss: 33299.2148 - accuracy: 0.4439 - val_loss: 32455.0000 - val_accuracy: 0.0091
Epoch 10/10
400/400 [==============================] - 2s 4ms/step - loss: 39299.6367 - accuracy: 0.4398 - val_loss: 38873.2891 - val_accuracy: 0.9731
<keras.callbacks.History at 0x7f1498c15f90>
```

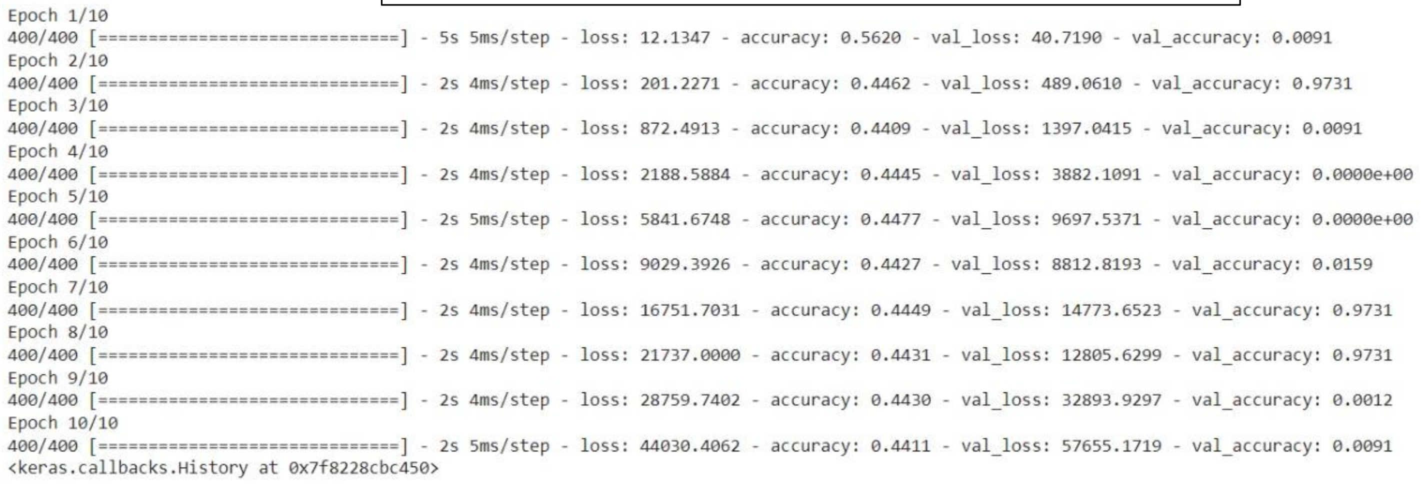
Fig : Result when keras tokenizer was used with preprocessing the data on jigsaw dataset

The main problem was that we were getting huge loss and low accuracy. The same happened when we used the bert model for tokenizing along with a LSTM model.



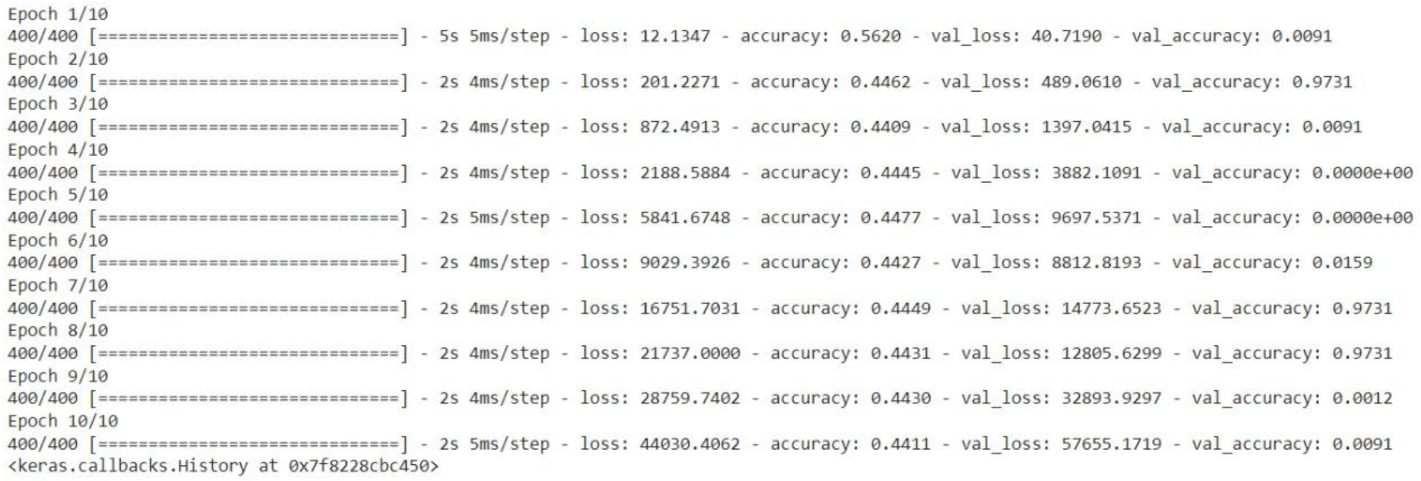
```
Epoch 1/10
400/400 [==============================] - 6s 7ms/step - loss: 7.2433 - accuracy: 0.5943 - val_loss: 18.7036 - val_accuracy: 0.0159
Epoch 2/10
400/400 [==============================] - 3s 8ms/step - loss: 185.5737 - accuracy: 0.4436 - val_loss: 327.5692 - val_accuracy: 0.9731
Epoch 3/10
400/400 [==============================] - 2s 5ms/step - loss: 708.4815 - accuracy: 0.4427 - val_loss: 1446.2972 - val_accuracy: 0.0159
Epoch 4/10
400/400 [==============================] - 2s 4ms/step - loss: 2410.4155 - accuracy: 0.4383 - val_loss: 2766.4741 - val_accuracy: 0.9731
Epoch 5/10
400/400 [==============================] - 2s 4ms/step - loss: 4940.5381 - accuracy: 0.4439 - val_loss: 5385.2192 - val_accuracy: 0.0091
Epoch 6/10
400/400 [==============================] - 2s 4ms/step - loss: 8708.2949 - accuracy: 0.4445 - val_loss: 11186.5977 - val_accuracy: 0.9731
Epoch 7/10
400/400 [==============================] - 2s 4ms/step - loss: 15321.6055 - accuracy: 0.4371 - val_loss: 11895.6846 - val_accuracy: 0.9731
Epoch 8/10
400/400 [==============================] - 2s 4ms/step - loss: 20472.6738 - accuracy: 0.4466 - val_loss: 50527.1289 - val_accuracy: 0.0012
Epoch 9/10
400/400 [==============================] - 2s 4ms/step - loss: 33299.2148 - accuracy: 0.4439 - val_loss: 32455.0000 - val_accuracy: 0.0091
Epoch 10/10
400/400 [==============================] - 2s 4ms/step - loss: 39299.6367 - accuracy: 0.4398 - val_loss: 38873.2891 - val_accuracy: 0.9731
<keras.callbacks.History at 0x7f1498c15f90>
```

Fig : Result when BERT with LSTM model was used without preprocessing the data on jigsaw dataset



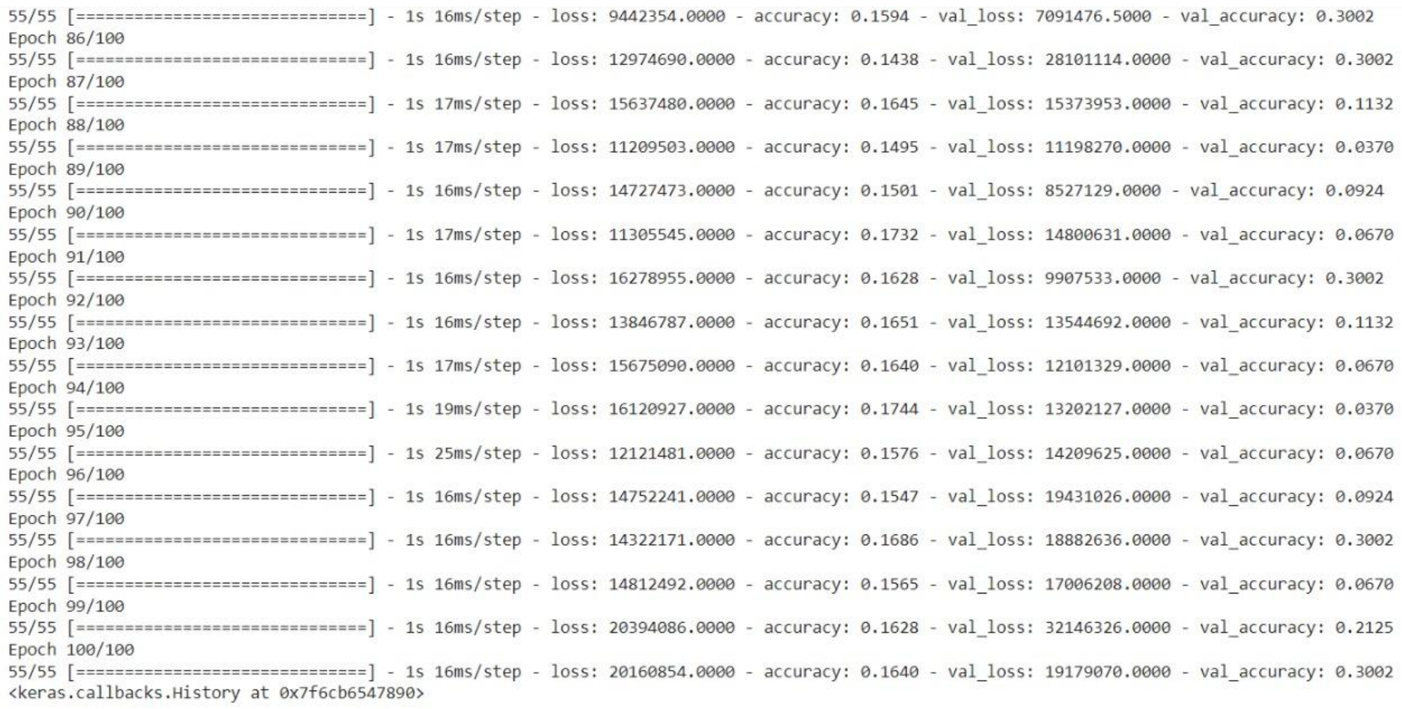
```
Epoch 1/10
400/400 [==============================] - 5s 5ms/step - loss: 12.1347 - accuracy: 0.5620 - val_loss: 40.7190 - val_accuracy: 0.0091
Epoch 2/10
400/400 [==============================] - 2s 4ms/step - loss: 201.2271 - accuracy: 0.4462 - val_loss: 489.0610 - val_accuracy: 0.9731
Epoch 3/10
400/400 [==============================] - 2s 4ms/step - loss: 872.4913 - accuracy: 0.4409 - val_loss: 1397.0415 - val_accuracy: 0.0091
Epoch 4/10
400/400 [==============================] - 2s 4ms/step - loss: 2188.5884 - accuracy: 0.4445 - val_loss: 3882.1091 - val_accuracy: 0.0000e+00
Epoch 5/10
400/400 [==============================] - 2s 5ms/step - loss: 5841.6748 - accuracy: 0.4477 - val_loss: 9697.5371 - val_accuracy: 0.0000e+00
Epoch 6/10
400/400 [==============================] - 2s 4ms/step - loss: 9029.3926 - accuracy: 0.4427 - val_loss: 8812.8193 - val_accuracy: 0.0159
Epoch 7/10
400/400 [==============================] - 2s 4ms/step - loss: 16751.7031 - accuracy: 0.4449 - val_loss: 14773.6523 - val_accuracy: 0.9731
Epoch 8/10
400/400 [==============================] - 2s 4ms/step - loss: 21737.0000 - accuracy: 0.4431 - val_loss: 12805.6299 - val_accuracy: 0.9731
Epoch 9/10
400/400 [==============================] - 2s 4ms/step - loss: 28759.7402 - accuracy: 0.4430 - val_loss: 32893.9297 - val_accuracy: 0.0012
Epoch 10/10
400/400 [==============================] - 2s 5ms/step - loss: 44030.4062 - accuracy: 0.4411 - val_loss: 57655.1719 - val_accuracy: 0.0091
<keras.callbacks.History at 0x7f8228cbc450>
```

Fig : Result when BERT with LSTM model was used with preprocessing the data on jigsaw dataset



```
Epoch 1/10
400/400 [==============================] - 5s 5ms/step - loss: 12.1347 - accuracy: 0.5620 - val_loss: 40.7190 - val_accuracy: 0.0091
Epoch 2/10
400/400 [==============================] - 2s 4ms/step - loss: 201.2271 - accuracy: 0.4462 - val_loss: 489.0610 - val_accuracy: 0.9731
Epoch 3/10
400/400 [==============================] - 2s 4ms/step - loss: 872.4913 - accuracy: 0.4409 - val_loss: 1397.0415 - val_accuracy: 0.0091
Epoch 4/10
400/400 [==============================] - 2s 4ms/step - loss: 2188.5884 - accuracy: 0.4445 - val_loss: 3882.1091 - val_accuracy: 0.0000e+00
Epoch 5/10
400/400 [==============================] - 2s 5ms/step - loss: 5841.6748 - accuracy: 0.4477 - val_loss: 9697.5371 - val_accuracy: 0.0000e+00
Epoch 6/10
400/400 [==============================] - 2s 4ms/step - loss: 9029.3926 - accuracy: 0.4427 - val_loss: 8812.8193 - val_accuracy: 0.0159
Epoch 7/10
400/400 [==============================] - 2s 4ms/step - loss: 16751.7031 - accuracy: 0.4449 - val_loss: 14773.6523 - val_accuracy: 0.9731
Epoch 8/10
400/400 [==============================] - 2s 4ms/step - loss: 21737.0000 - accuracy: 0.4431 - val_loss: 12805.6299 - val_accuracy: 0.9731
Epoch 9/10
400/400 [==============================] - 2s 4ms/step - loss: 28759.7402 - accuracy: 0.4430 - val_loss: 32893.9297 - val_accuracy: 0.0012
Epoch 10/10
400/400 [==============================] - 2s 5ms/step - loss: 44030.4062 - accuracy: 0.4411 - val_loss: 57655.1719 - val_accuracy: 0.0091
<keras.callbacks.History at 0x7f8228cbc450>
```

Fig : Result when BERT with LSTM model was used without preprocessing the data on ethos dataset



```
55/55 [==============================] - 1s 16ms/step - loss: 9442354.0000 - accuracy: 0.1594 - val_loss: 7091476.5000 - val_accuracy: 0.3002
Epoch 86/100
55/55 [==============================] - 1s 16ms/step - loss: 12974690.0000 - accuracy: 0.1438 - val_loss: 28101114.0000 - val_accuracy: 0.3002
Epoch 87/100
55/55 [==============================] - 1s 17ms/step - loss: 15637480.0000 - accuracy: 0.1645 - val_loss: 15373953.0000 - val_accuracy: 0.1132
Epoch 88/100
55/55 [==============================] - 1s 17ms/step - loss: 11209503.0000 - accuracy: 0.1495 - val_loss: 11198270.0000 - val_accuracy: 0.0370
Epoch 89/100
55/55 [==============================] - 1s 16ms/step - loss: 14727473.0000 - accuracy: 0.1501 - val_loss: 8527129.0000 - val_accuracy: 0.0924
Epoch 90/100
55/55 [==============================] - 1s 17ms/step - loss: 11305545.0000 - accuracy: 0.1732 - val_loss: 14800631.0000 - val_accuracy: 0.0670
Epoch 91/100
55/55 [==============================] - 1s 16ms/step - loss: 16278955.0000 - accuracy: 0.1628 - val_loss: 9907533.0000 - val_accuracy: 0.3002
Epoch 92/100
55/55 [==============================] - 1s 16ms/step - loss: 13846787.0000 - accuracy: 0.1651 - val_loss: 13544692.0000 - val_accuracy: 0.1132
Epoch 93/100
55/55 [==============================] - 1s 17ms/step - loss: 15675090.0000 - accuracy: 0.1640 - val_loss: 12101329.0000 - val_accuracy: 0.0670
Epoch 94/100
55/55 [==============================] - 1s 19ms/step - loss: 16120927.0000 - accuracy: 0.1744 - val_loss: 13202127.0000 - val_accuracy: 0.0370
Epoch 95/100
55/55 [==============================] - 1s 25ms/step - loss: 12121481.0000 - accuracy: 0.1576 - val_loss: 14209625.0000 - val_accuracy: 0.0670
Epoch 96/100
55/55 [==============================] - 1s 16ms/step - loss: 14752241.0000 - accuracy: 0.1547 - val_loss: 19431026.0000 - val_accuracy: 0.0924
Epoch 97/100
55/55 [==============================] - 1s 16ms/step - loss: 14322171.0000 - accuracy: 0.1686 - val_loss: 18882636.0000 - val_accuracy: 0.3002
Epoch 98/100
55/55 [==============================] - 1s 16ms/step - loss: 14812492.0000 - accuracy: 0.1565 - val_loss: 17006208.0000 - val_accuracy: 0.0670
Epoch 99/100
55/55 [==============================] - 1s 16ms/step - loss: 20394086.0000 - accuracy: 0.1628 - val_loss: 32146326.0000 - val_accuracy: 0.2125
Epoch 100/100
55/55 [==============================] - 1s 16ms/step - loss: 20160854.0000 - accuracy: 0.1640 - val_loss: 19179070.0000 - val_accuracy: 0.3002
<keras.callbacks.History at 0x7f6cb6547890>
```

Fig : Result when BERT with LSTM model was used with preprocessing the data on ethos dataset

Future Improvisation

For Future modifications, we can use text augmentation methods to increase the size of the Ethos Dataset. The text augmentation methods include :

- Back Translation - This is one of the simplest text augmentation methods in which we translate the data into a foreign language and again translate it to the original language. So, as the output you get a different sentence with the same meaning.
- Easy Data Augmentation - This is a method in which we randomly choose a word and replace it with its synonyms or multiple words are chosen and replaced in the sentence.

- NLP Aug Library - This is a library which offers multiple methods for data augmentation like character level augmentation, word level augmentation and Sentence level augmentation.

Conclusion

In this report, our main target was to experiment with different feature selection and input representation techniques with classical machine learning algorithms for the classification of multi-label hate speech. We have tried to tackle this classification task with variants of LSTM classifiers. A few of the most recent related works have shown that unconventional input representation such as word-embeddings could be useful for this task even with the conventional classifiers. So far, we have not seen much research done on multi label hate speech. So our prime aspirations were to experiment with these features and input representations.

We also observed that hate speech detection is a difficult task to accomplish, and is data-driven. We also saw that not all the datasets could help achieve good results, specially because of the nature of the datasets and annotation schemes. Hate speech also heavily depends on the culture and location of the text being encountered. Current study shows that classical ML approach with fine-tuned feature engineering could compete with state-of-the-art deep neural network-based models and in many ways are better. Our findings also support this hypothesis.

In a nutshell, our efforts in this report can be seen as :

- We have tried to implement ML models like LSTM and Bi-LSTM for the task of multi-label (violence, gender, race etc.) hate speech classification.
- We have experimented with multiple text vectorization models like BERT, GloVE, Keras Tokenizer.

We want to conclude our report with an open question for the readers -

“ Are all the classifiers good for Multi-Label Hate Speech Detection ?
Or is it just the way we feed the data to them ?”

References

[1] https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0221152

[2] https://link.springer.com/article/10.1007/s40747-021-00608-2

[3] https://paperswithcode.com/dataset/ethos

[4] https://www.kaggle.com/competitions/jigsaw-toxic-comment-classification-challenge/data

[5] https://www.mdpi.com/2076-3417/10/23/8614

[6] https://ijirt.org/master/publishedpaper/IJIRT149275_PAPER.pdf

[7] https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9455353