FLIP ROBO

Malignant comments multiclass classification DL-ML Model

Submitted by:

Ranjit Maity

ACKNOWLEDGMENT

I would like to express my special thanks of gratitude to FLIP ROBO TECHNOLOGIES. as well as our SME Swati Rustagi who gave me the golden opportunity to do this wonderful project on the topic Project Malignant Comment Classification which also helped me in doing this project.

INTRODUCTION

- Business Problem Framing

- The proliferation of social media enables people to express their opinions widely online. However, at the same time, this has resulted in the emergence of conflict and hate, making online environments uninviting for users. Although researchers have found that hate is a problem across multiple platforms, there is a lack of models for online hate detection.
- Online hate, described as abusive language, aggression, cyberbullying, hatefulness and many others has been identified as a major threat on online social media platforms. Social media platforms are the most prominent grounds for such toxic behaviour.

- There has been a remarkable increase in the cases of cyberbullying and trolls on various social media platforms. Many celebrities and influences are facing backlashes from people and have to come across hateful and offensive comments. This can take a toll on anyone and affect them mentally leading to depression, mental illness, self-hatred and suicidal thoughts.

Solution is:-

We will build a prototype of online hate and abuse comment classifier which can used to classify hate and offensive comments so that it can be controlled and restricted from spreading hatred and cyberbullying.

From this model we can be predict whether the comment is 'malignant','highly_malignant','rude','threat','abuse','loathe'.

Conceptual Background of the Domain Problem

Internet comments are bastions of hatred and vitriol. While online anonymity has provided a new outlet for aggression and hate speech, machine learning can be used to fight it. The problem we sought to solve was the tagging of internet comments that are aggressive towards other users. This means that insults to third parties such as celebrities will be tagged as unoffensive, but "u are an idiot" is clearly offensive.

- Review of Literature

https://medium.com/@nupurbaghel/toxic-comment-classification-f6e075c3487a

- Motivation for the Problem Undertaken

I have seen victims of such cases get into depression. I always wanted to help such people but I felt helpless but now I have a way to help them.

Reference:-

https://medium.com/@nupurbaghel/toxic-comment-classification-f6e075c3487a

Analytical Problem Framing

- Mathematical/ Analytical Modelling of the Problem

  Removing excessive length for training data here.
  Some very large length comments can be seen, in our dataset. These pose serious problems like adding excessively more words to the training dataset, causing training time to increase and accuracy to decrease.
  Hence, a **threshold of 400 characters** will be created and only comments which have length smaller than 400 will be used further.

- Data Sources and their formats

  Primarily train data and test data both were in .csv file format with no missing values in the data set.

The dataset shape is Train.csv→ `(159571, 8)`
Test.csv→ `(153164, 2)`

Data Pre-processing Done

### Removing excessive length for training data only.(kept as 400 keeping the computation power in consideration.)

1. Initializing the Vocabulary size(5000)

2. Making of corpus by GENERIC NLP FILTER CODE

a. Convert all cases to lower

b. Remove punctuations

c. Remove Stopwords

d. Stemming and Lemmatising

3.Applying embedding technique.(Count Vectoriser is giving memory error)

4. OneHot Encoding(or dummies)

# **Initializing maximum length of embedding_vector_features=30.**

5. Applying embedding technique.(Count Vectoriser is giving memory error)

embedded docs--numerical(vector) format can be input to any ML or DL model)

6. Train_test_split training data(embedded_docs)in a different way as our target feature('labels') contains 6 classes.

#Our data is now ready for model building process input i.e vectors(numerical format).

- Data Inputs- Logic- Output Relationships

The DL-NN and SVM classifier algorithm Used here is is a popular result oriented classifier algorithm.

Save the DL model →.h5 format

Save the ML model→.obj format

- State the set of assumptions (if any) related to the problem under consideration

  Sentence length are reduced to 400.

- Hardware and Software Requirements and Tools Used

  Hardware-64bit, 12GB RAM, 240GB SSD.
  Software-Excel, Anaconda,jupyter notebook,python 3.6

  ## Libraries used:-

  1. numpy
  2. pandas
  3. matplotlib
  4. sweetviz
  5. seaborn
  6. sklearn
  7. itertools
  8. Seaborn
  9.Tensorflow.keras
  10. tensorflow.keras.preprocessing.text

Model/s Development and Evaluation

- Identification of possible problem-solving approaches (methods)
  - **Sentence length are reduced to 400.**

    Scores improved drastically after this approach for DL-NN model.

    And the Model computation cost was reduced.

- Testing of Identified Approaches (Algorithms)

1.MultinomialNB

2.BinaryRelevance

3.SVM

4.GaussianNB

5.ClassifierChain

6.LabelPowerset

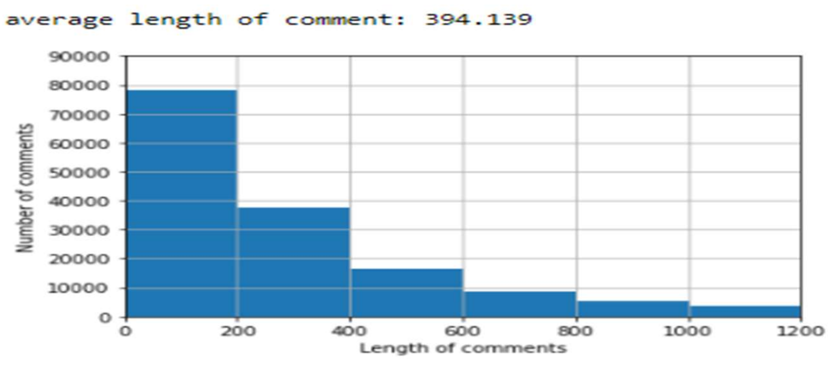
7.MLkNN

Key Metrics for success in solving problem under consideration.

This is a text multiclass classification problem and I have choosen
# 1.Accuracy,
# 2.Log-loss,
# 3.Hammingloss as my evaluation metrics

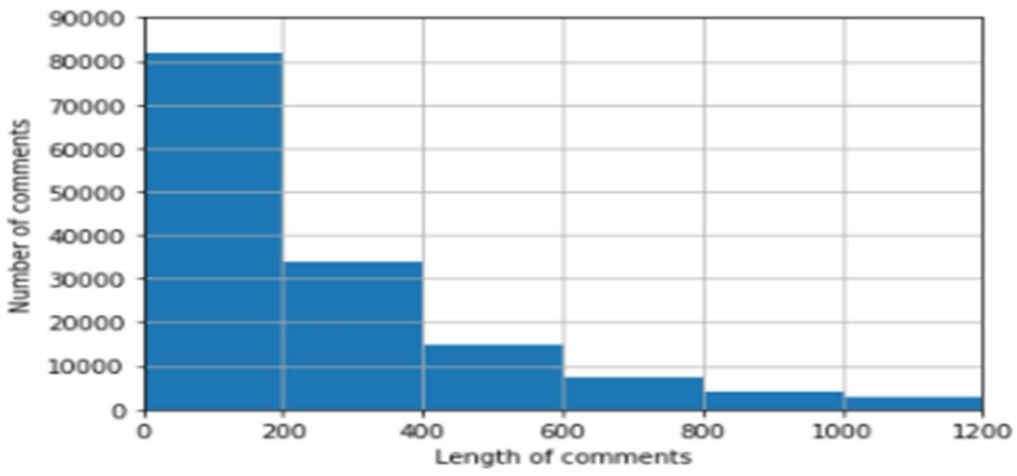
Visualizations

1.Analyzing No. of comments for **train data** having lengths varying from 0 to 1200

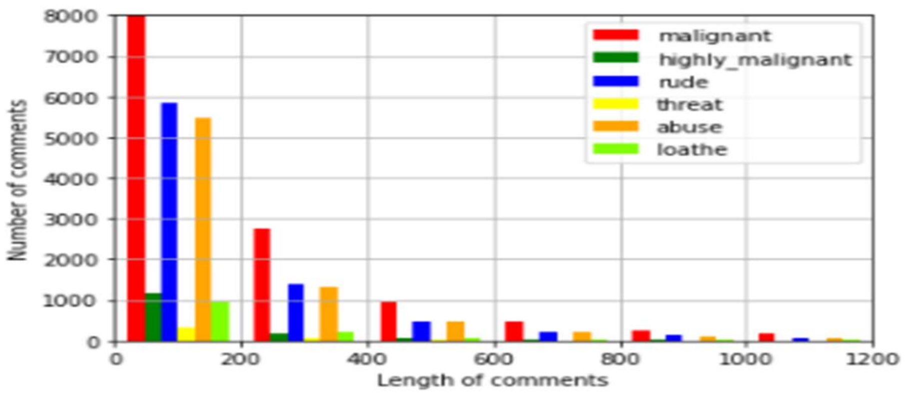


2.Analyzing No. of comments for **test data** having lengths varying from 0 to 1200

```
average length of comment: 364.875
```

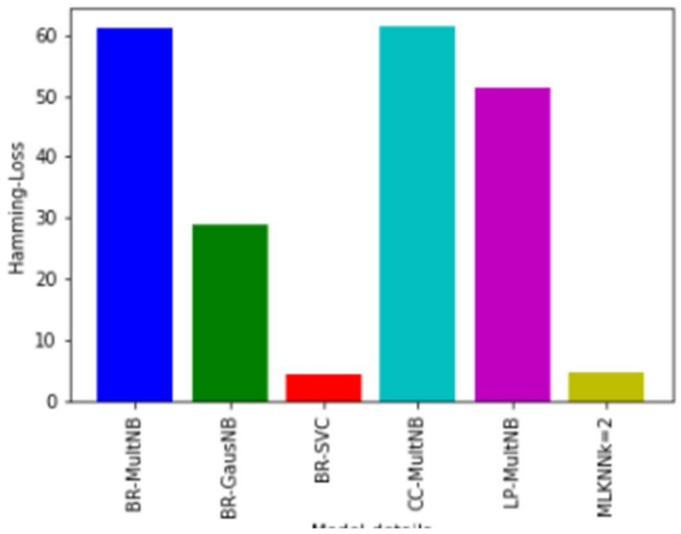


3. Number of comments classified as malignant,highly_malignant,rude,....etc depending on their lengths for train data.
Inference from Data-→Data with low sentence lengths(<200) are malignant,highly_malignant,rude.

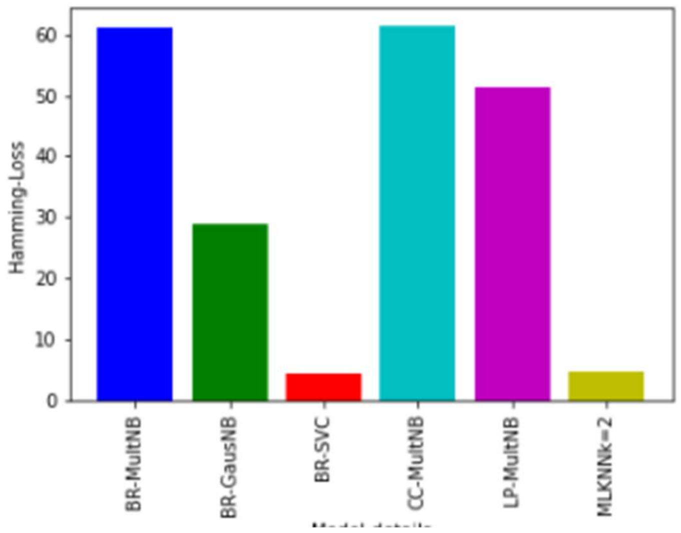


Some very large length comments can be seen, in our dataset. These pose serious problems like adding excessively more words to the training dataset, causing training time to increase and accuracy to decrease! Hence, a threshold of 400 characters will be created and only comments which have length smaller than 400 will be used further.

4.Hamming loss comparision of all models



5.Log-loss of all models



- Interpretation of the Results

  Some very large length comments can be seen, in our dataset. These pose serious problems like adding excessively more words to the training dataset, causing training time to increase and accuracy to decrease Hence, a threshold of 400 characters will be created and only comments which have length smaller than 400 will be used further.

Pre-processing:-

# 1. Import necessary libraries

# 2.Read the train.csv and test.csv file(parallely preprocessing is done for both train and test datasets).

# 3.Shuffling of indices, to avoid using train_test_split later
df = df.reindex(np.random.permutation(df.index))

# 4. Separate the comment feature(X) data and outcome labels(y)
#label = df[['malignant','highly_malignant','rude','threat','abuse','loathe']]

# Visualizations

# Data Preprocessing
### Removing excessive length for training data only.(kept as 400 keeping the computation power in consideration.)

# 5. Initializing the Vocabulary size(5000)

# 6. Making of corpus by GENERIC NLP FILTER CODE

a. Convert all cases to lower
b. Remove punctuations
c. Remove Stopwords
d. Stemming and Lemmatising

# 7.Applying embedding technique.(Count Vectoriser is giving memory error)

# 8. OneHot Encoding(or dummies)

# Initializing maximumu length of embedding_vector_features=30.

# 9. Applying embedding technique.(Count Vectoriser is giving memory error)
embedded docs--numerical(vector) format can be input to any ML or DL model)

# 10. Train_test_split training data(embedded_docs)in a different way as our target feature('labels') contains 6 classes.

#Our data is now ready for model building process input numerical format.

# Define all the 3 evaluation metrics(accuracy,logloss,hammingloss)

# PART-1-DL-NN-Model Building(score=87.97%) after GridsearchCV(88.007%)

.X(embedded_docs) y(df['label])
. Model training(20 epochs)
. Predicting
. Check the accuracy. (accuracy=87.96%)

# GridsearchCV.(accuracy=88.26%)

# PART-2--Multiple classifier ML models(4types)
# X(embedded_docs) y(df['label])

# A. BinaryRelevance

from sklearn.naive_bayes import MultinomialNB

from sklearn.svm import SVC

from sklearn.naive_bayes import GaussianNB

# B.ClassifierChain

from skmultilearn.problem_transform import ClassifierChain

# C. LabelPowerset

from skmultilearn.problem_transform import LabelPowerset

# D.MLKNN

from skmultilearn.adapt import MLkNN

#In all the Models above
11. Predicting
12. Check the accuracy,log-loss,hamming score.

# Visualzitaion of Log-Loss and hamming loss of all models.

# Comparing All models and Concluding!

On basis of **hamming loss:-------------------
The best model would be **SVM classifier**. It has lowest hamming-loss of 4.384 % only.

On basis of **Log-Loss:--------------------
The best model will be the **Neural Network Model**. It has lowest Log-loss of 0.375 % only.

CONCLUSION

- Key Findings and Conclusions of the Study

→Data with low sentence lengths(<200) are malignant,rude,abuse.

→Some very large length comments can be seen, in our dataset. These pose serious problems like adding excessively more words to the training dataset, causing training time to increase and accuracy to decrease.

- Learning Outcomes of the Study in respect of Data Science

New things I implemented here in the Part-2 Machine Learning model building in this notebook for Multiclass classification where the predicted output has 6 different classes all together predicting in a single shot. Below are the listed new Machine Learning model that I Leart in this project.

A. BinaryRelevance

from sklearn.naive_bayes import MultinomialNB

from sklearn.svm import SVC

from sklearn.naive_bayes import GaussianNB

B.ClassifierChain

from skmultilearn.problem_transform import ClassifierChain

C. LabelPowerset

from skmultilearn.problem_transform import LabelPowerset

D.MLKNN

from skmultilearn.adapt import MLkNN

Balancing the data by Upsampling technique (RandomOverSampler).

- Limitations of this work. And Scope for Future Work

  Computational complexity:

  Model building and GridsearchCV takes too much time for algorithms like DL-NN,KNN and SVC.

Hardware problem:-

Need more powerful system.

My maximum time went in GridsearchCV(DL-NN) and ML (SVC,MLKNN)model building part.

Problems I faced during project

BR-SVC algorithm took lot of time(>4-5hours)

DL-NN Gridsearchcv took lot of time(>4-8hours)

I could not play around the values while hypertuning due to low computational power. MY accuracy after GridsearchCV increased just a little bit.

Not upto my expectations.More work to be done in GridsearchCV.

Maximum time went in computing rather than analysis of the data.

Future works(A project is always developing)

1. Analysing the graphs more deeply.

2. Try different parameters combinations for hypertuning(part-1-GridsearchCV for DL-NN model).