FLIP ROBO

Malignant Comments Classification

Submitted by:

Amey Takle

ACKNOWLEDGMENT

- **ToxicComment ClassificationUsing Neural Networksand Machine Learning** [Revati Sharma1, Meetkumar Patel2]
- **Blog by Shaunak Varudandi on towards data science names Toxic Comment Classification using LSTM and LSTM-CNN and Deployment using Amazon AWS EC2**
- **Deep Learning for Hate Speech Detection in Tweets** [Pinkesh Badjatiya (IIIT-H) , Shashank Gupta (IIIT-H), Manish Gupta (Microsoft), Vasudeva Varma (IIIT-H)]
- **Expressively Vulgar: The Socio-dynamics of Vulgarity** [University of Texas at Austin, Isabela Cachola, Eric Holgate, and Junyi Jessy Li. From the University of Pennsylvania, Daniel Preotiuc-Pietro]
- **Multilingual Twitter Sentiment Classification: The Role of Human Annotators** [Igor Mozetič, Miha Grčar, and Jasmina Smailovič, from the Department of Knowledge Technologies at the Jožef Stefan Institute]

INTRODUCTION

- Business Problem Framing

  Business requirement in this problem was to classify the given train and test dataset comments into given classes.

  The implementation will help to segregate the comments on social media when implemented with the sites.

- Conceptual Background of the Domain Problem

  Classification and sentiment analysis will play a key role in obtaining a solution for the problem.

  A study of frequently occurring words in the corpus will help to get an idea of the different classes.

- Motivation for the Problem Undertaken

  The implementation of the project will help to control toxicity on social media.

Analytical Problem Framing

- Data Sources and their formats

  Data for this project can be pulled from various social network sites.

  Comments on various posts can be scraped to create corpus.

- Data Preprocessing Done

  Stop words cleaning

  Symbols cleaning was also done.

  Irrelevant columns like IDs were dropped.

- Data Inputs- Logic- Output Relationships

  Emotions from text can be extracted based on frequently occurring words.

  Second approach is to create a common class which represents all the malignant classes and other class which is normal class.

- Hardware and Software Requirements and Tools Used

  The software development was done on jupyter notebook
  Hardware specs used were:

  1. 8 GB memory
  2. I7 processor
  3. Os: windows 10 home 2019

Model/s Development and Evaluation

- Identification of possible problem-solving approaches (methods)

  Two approaches that were followed are as follows:

  1. Sentiment Analysis
     Frequently occurring words from the classes were extracted and test data was searched for those words to classify/ attach that particular emotion to text
  2. Classification
     The train data was broadly classified into 2 classes, malignat and normal comments

- Testing of Identified Approaches (Algorithms)
  1. Naïve Bayes Classifier
  2. Decision Tree Classifier
  3. Random Forest Classifier

- Run and Evaluate selected models

Approach 1:

1. Frequently occurring words from the classes were extracted
2. data was searched for those words to classify/ attach that particular emotion to text

```
In [19]: emotion_list=[]
with open('emotions.txt','r') as file:
    for line in file:
        clear_line=line.replace('\n','').replace(',','').replace("'",'').strip()
        word,emotion=clear_line.split(':')

        if word in ds_train['cleaned_comments'][42]:
            emotion_list.append(emotion)
print('emotions from text\n',emotion_list,'\n\n')
w=Counter(emotion_list)
print('counter of each emotion \n',w)

emotions from text
 ['malignant', 'malignant', 'malignant', 'malignant', 'malignant', 'highly_malignant', 'highly_malignant', 'hi
ghly_malignant', 'highly_malignant', 'highly_malignant', 'rude', 'rude', 'rude', 'rude', 'threat', 'threat', '
threat', 'abuse', 'abuse', 'abuse', 'loathe', 'loathe', 'loathe', 'loathe']


counter of each emotion
 Counter({'malignant': 5, 'highly_malignant': 5, 'rude': 4, 'loathe': 4, 'threat': 3, 'abuse': 3})
```

```
In [20]: op_dict={}
for comment in ds_test['comment_text']:
    emotion_list=[]
    with open('emotions.txt','r') as file:
        for line in file:
            clear_line=line.replace('\n','').replace(',','').replace("'",'').strip()
            word,emotion=clear_line.split(':')

            if word in comment.split():
                emotion_list.append(emotion)
    print('emotions from text\n',emotion_list,'\n\n')
    w=Counter(emotion_list)
    print('counter of each emotion \n',w)
    op_dict[comment]=emotion_list

emotions from text
 ['malignant', 'highly_malignant', 'highly_malignant', 'rude', 'threat', 'threat', 'threat', 'abuse', 'abuse
', 'loathe', 'loathe']


counter of each emotion
 Counter({'threat': 3, 'highly_malignant': 2, 'abuse': 2, 'loathe': 2, 'malignant': 1, 'rude': 1})
emotions from text
 []


counter of each emotion
 Counter()
emotions from text
 []


counter of each emotion
 Counter()
```

op_dict is the output dictionary which list all the comments in test data with the corresponding emotion in that comment.

If the value is an empty list that represents that the comment is normal.

Approach 2:

1. The comments in train were broadly classified as either malignant or normal and a new column called comment_cat was added to data set

	comment_text	malignant	highly_malignant	rude	threat	abuse	loathe	cleaned_comments	comment_cat
0	explanation why the edits made under my userna...	0	0	0	0	0	0	explanation edits made username hardcore metal...	0
1	d aww he matches this background colour i m se...	0	0	0	0	0	0	aww matches background colour seemingly stuck ...	0
2	hey man i m really not trying to edit war it s...	0	0	0	0	0	0	hey man really trying edit war guy constantly ...	0
3	more i can t make any real suggestions on imp...	0	0	0	0	0	0	make real suggestions improvement wondered sec...	0
4	you sir are my hero any chance you remember wh...	0	0	0	0	0	0	sir hero chance remember page	0
...	...	...	...	...	...	...	...	...	...

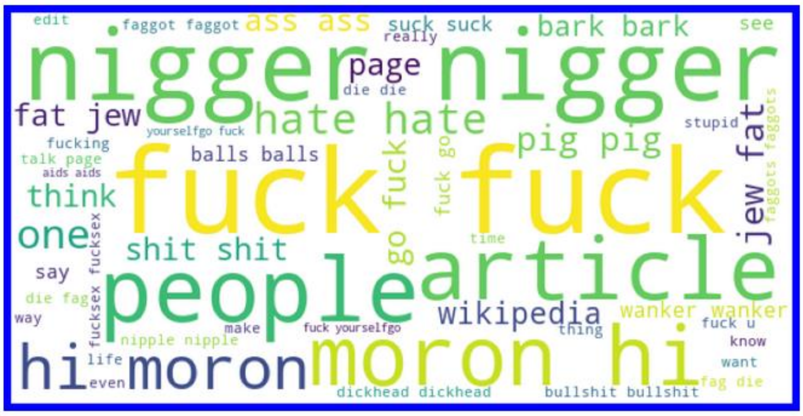
Further classification algorithms were trained according to the train data set and predictions on test data set were done

- Key Metrics for success in solving problem under consideration
  Metrics that were used were as follows:
  1. accuracy_score
  2. classification_report
  3. confusion_matrix

- **Visualizations**

First visualization that was used here was a word cloud

```
In [11]: word_count(malignant)
```




```
['fuck fuck', 'nigger nigger', 'people', 'article', 'moron hi', 'hi moron', 'hate hate', 'one', 'pig pig', 'je
w fat', 'fat jew', 'page', 'think', 'shit shit', 'go fuck', 'ass ass', 'wikipedia', 'bark bark', 'wanker wanke
r', 'say', 'suck suck', 'see', 'fuck go', 'balls balls', 'bullshit bullshit', 'know', 'make', 'even', 'nipple
nipple', 'faggot faggot', 'want', 'stupid', 'fucking', 'talk page', 'edit', 'life', 'time', 'way', 'fuck u', '
thing', 'die die', 'really', 'dickhead dickhead', 'die fag', 'fag die', 'faggots faggots', 'fucksex fucksex',
'fuck yourselfgo', 'yourselfgo fuck', 'aids aids']
```

To get the word clouds list of comments with different classes were created.

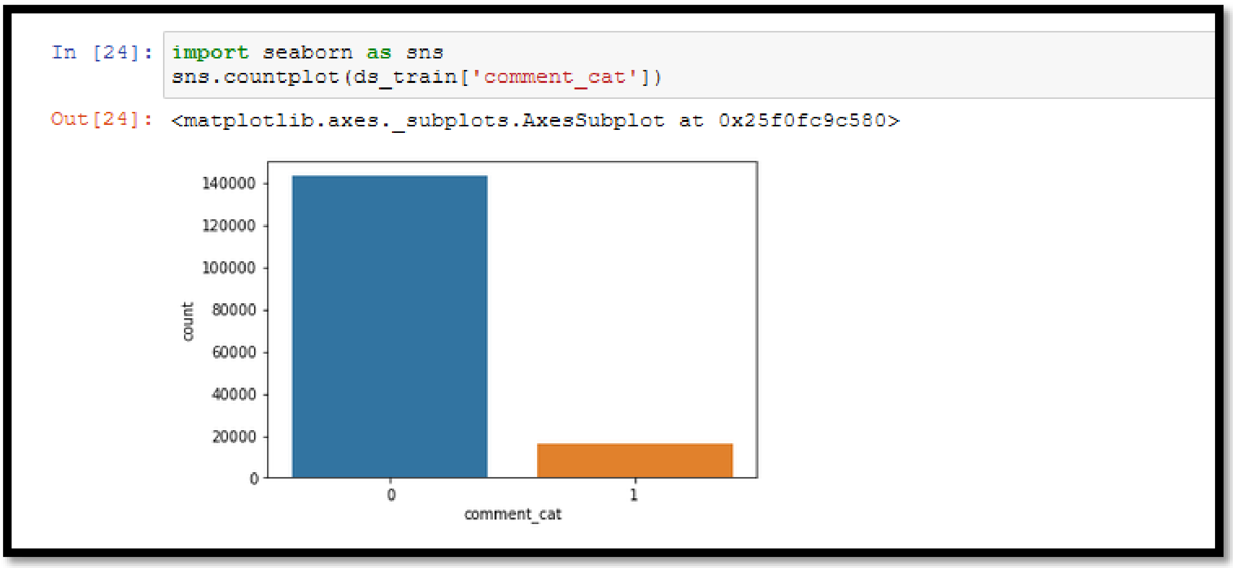
So the created lists were as follows:

1. Malignant
2. highly_malignant
3. rude
4. threat
5. abuse
6. loathe

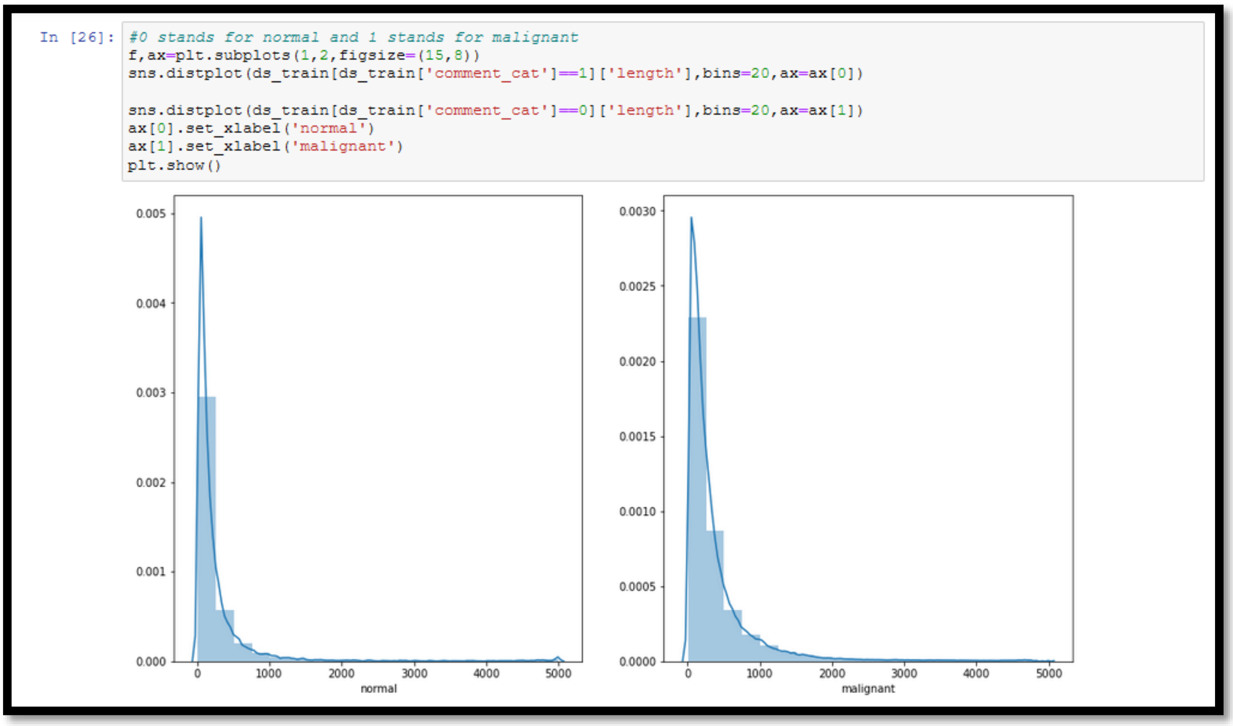
the word clouds for these list were then plotted accordingly.

The comments were then classified in 2 classes and a column comment_cat was added

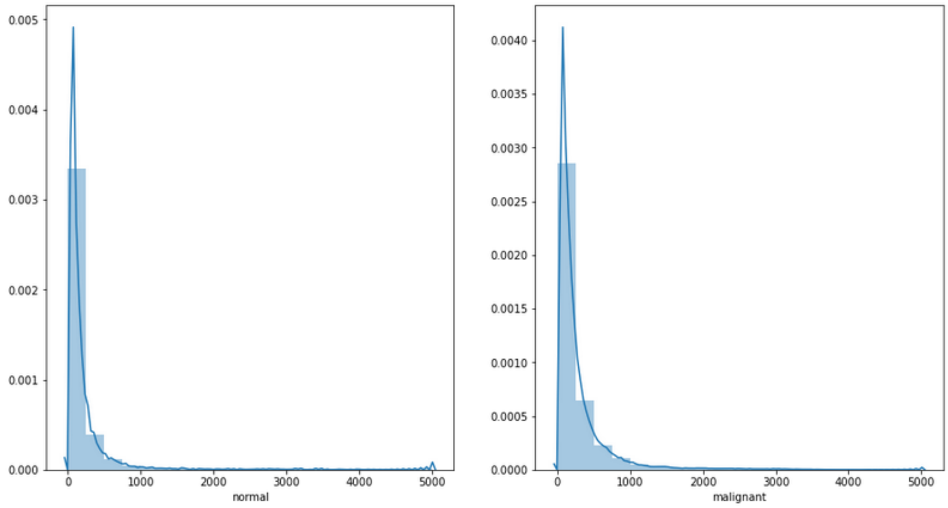
Next visualization was a count plot of this column



Further the actual length and cleaned length were plotted to get an idea of stopwrods in the comments



In [27]:
```
f,ax=plt.subplots(1,2,figsize=(15,8))
sns.distplot(ds_train[ds_train['comment_cat']==1]['cleaned_length'],bins=20,ax=ax[0])

sns.distplot(ds_train[ds_train['comment_cat']==0]['cleaned_length'],bins=20,ax=ax[1])
ax[0].set_xlabel('normal')
ax[1].set_xlabel('malignant')
plt.show()
```



Further visualization were for the results of diff models

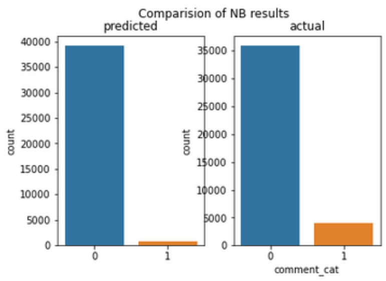
- Interpretation of the Results
    1. The data set is imbalanced
    2. Very low malignant comments are present in the given data
    3. There are some frequently occurring words that help to differentiate the classes.
    4. Plotting the distribution of actual length and cleaned lengths helps to get an idea of the stop words in the comments

RESULTS

- **Naïve Bayes:**

```
In [32]: print(classification_report(y_test,y_pred_nb))
         print(confusion_matrix(y_test,y_pred_nb))

                  precision    recall  f1-score   support

               0       0.92      1.00      0.96     35819
               1       0.99      0.19      0.32      4068

        accuracy                           0.92     39887
       macro avg       0.95      0.59      0.64     39887
    weighted avg       0.92      0.92      0.89     39887

[[35810     9]
 [ 3299   769]]
```

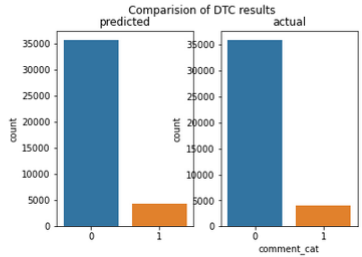
```
In [33]: plt.subplot(1, 2, 1)
         sns.countplot(y_pred_nb)
         plt.title('predicted')
         plt.subplot(1, 2, 2)
         sns.countplot(y_test)
         plt.title('actual')
         plt.suptitle('Comparision of NB results')

Out[33]: Text(0.5, 0.98, 'Comparision of NB results')
```



Naïve Bayes provided an accuracy of 0.9170, but very low actual malignant comments were classified correctly

- **Decision Tree Classifier:**

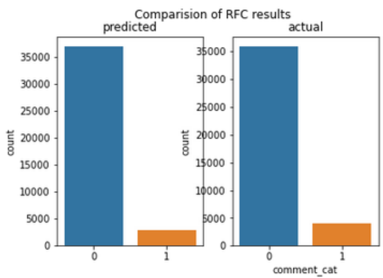
```
In [35]: print(classification_report(y_test,y_pred_dtc))
         print(confusion_matrix(y_test,y_pred_dtc))

              precision    recall  f1-score   support

           0       0.97      0.96      0.96     35819
           1       0.67      0.71      0.69      4068

    accuracy                           0.93     39887
   macro avg       0.82      0.84      0.83     39887
weighted avg       0.94      0.93      0.94     39887

[[34382  1437]
 [ 1168  2900]]

In [36]: plt.subplot(1, 2, 1)
         sns.countplot(y_pred_dtc)
         plt.title('predicted')
         plt.subplot(1, 2, 2)
         sns.countplot(y_test)
         plt.title('actual')
         plt.suptitle('Comparision of DTC results')

Out[36]: Text(0.5, 0.98, 'Comparision of DTC results')
```



Decision Tree Classifier provided an accuracy of 0.9346, DTC has classified a satisfactory amount of malignant comments correctly

- **Random forest classifier:**

```
In [38]: print(classification_report(y_test,y_pred_rfc))
         print(confusion_matrix(y_test,y_pred_rfc))

              precision    recall  f1-score   support

           0       0.95      0.98      0.97     35819
           1       0.81      0.58      0.68      4068

    accuracy                           0.94     39887
   macro avg       0.88      0.78      0.82     39887
weighted avg       0.94      0.94      0.94     39887

[[35277   542]
 [ 1712  2356]]

In [39]: plt.subplot(1, 2, 1)
         sns.countplot(y_pred_rfc)
         plt.title('predicted')
         plt.subplot(1, 2, 2)
         sns.countplot(y_test)
         plt.title('actual')

         plt.suptitle('Comparision of RFC results')

Out[39]: Text(0.5, 0.98, 'Comparision of RFC results')
```



Random forest classifier provided an accuracy of 0.9434, RFC has classified a satisfactory amount of malignant comments correctly but these are lesser than decision tree.

Looking at the ability to classify malignant comments DTC model was finalized and pickled

CONCLUSION

- Key Findings and Conclusions of the Study

  Sentiment analysis proves to be a better approach when dealing with multiple classes in text.

  Comparison of predicted and actual results help greatly to get an idea of model performance.

- Learning Outcomes of the Study in respect of Data Science

  Visualization of the frequently occurring words in text data greatly helps to get an idea of how a particular class might look and helps the model to better recognize the provided data.
  Data cleaning helps reduce the processing time in case of text data.
  Visualization of results must be done to get an idea of model performance.

- Limitations of this work and Scope for Future Work

  The emotions.txt file for this project can be improved so that more accurate emotions can be predicted form the given data.