FLIP ROBO

Malignant Comment Classifier Project

Submitted by:

Ashish Kumar Dewangan

1. Project Overview

Social media platforms are means to express ourselves for our audience. Any kind of malignant comment or expression by other individual can pose a serious threat or insult or harassment which can make the user uncomfortable. As a result of which the user may refrain himself or herself from using social media platforms in future. Hence, it is the responsibility of the social media platforms to identify and take appropriate actions to prevent any kind of harm to the user.

Currently there are 3.6 billion users out of total 7.9 billion who are engaged in one or the other form of social media platform. Which means that every one-in-two people uses social media platform. This problem thus can be eliminated as it falls under the category of Natural Language Processing. In this, we try to recognize the intention of the speaker by building a model that's capable of detecting different types of toxic, severe_toxic, obscene, threat, insult and/or identity_hate. Moreover, it is crucial to handle any such kind of nuisance, to make a more user-friendly experience, only after which people can actually enjoy in participating in discussions with regard to online conversation.

2. Problem Statement

The data has comments posted by a user and our task is to identify the malignant comment and categorise them as toxic, severe_toxic, obscene, threat, insult and/or identity_hate. Hence, this problem falls under category of Multi Label Classification Problem.

Multiple ways are there to approach this classification problem which are as follows:

- Multi-label methods which belong to the problem transformation category: Label Power Set (LP), Binary Relevance (BR), BR+ (BRplus), and classifier chain.

- Base and adapted algorithms: J48 (Decision Tree), Naïve Bayes, k-Nearest-Neighbor (KNN), SMO (Support Vector Machines), and, BP-MLL neural networks.

Further, out of the total dataset used for experimenting these algorithms, 70% was used for training and 30% was used for testing. Each testing dataset was labelled and thus for each algorithm using the predictions and labels, calculation of metric such as hamming-loss, accuracy and log-loss was done. The final results have been complied on the basis of values obtained by algorithmic models in hamming-loss and log-loss combined.

3. Data Exploration

The dataset consists of the following fields-

Out of these fields, the comment_text field will be preprocessed and fitted into different classifiers to predict whether it belongs to one or more of the labels/outcome variables (i.e. toxic, severe_toxic, obscene, threat, insult and identity_hate).

id	A unique id aligned with each comment text.
comment_text	It includes the comment text.
malignant	It is a column with binary values depicting which comments are malignant in nature.
highly_malignant	Binary column with labels for highly malignant text.
rude	Binary column with labels for comments that are rude in nature.
threat	Binary column with labels for threatening context in the comments.
abuse	Binary column with labels with abusive behaviour.

We have a total of 159571 samples of comments and labelled data, which can be loaded from train.csv file. One more very crucial aspect of the dataset is noticing the frequency occurrence of multi- labelled data.

Label	Frequency	Percentage
malignant	15294	9.6
highly_malignant	1595	1.0
rude	8449	5.3
threat	478	0.3
abuse	7877	4.9
loathe	1405	0.9

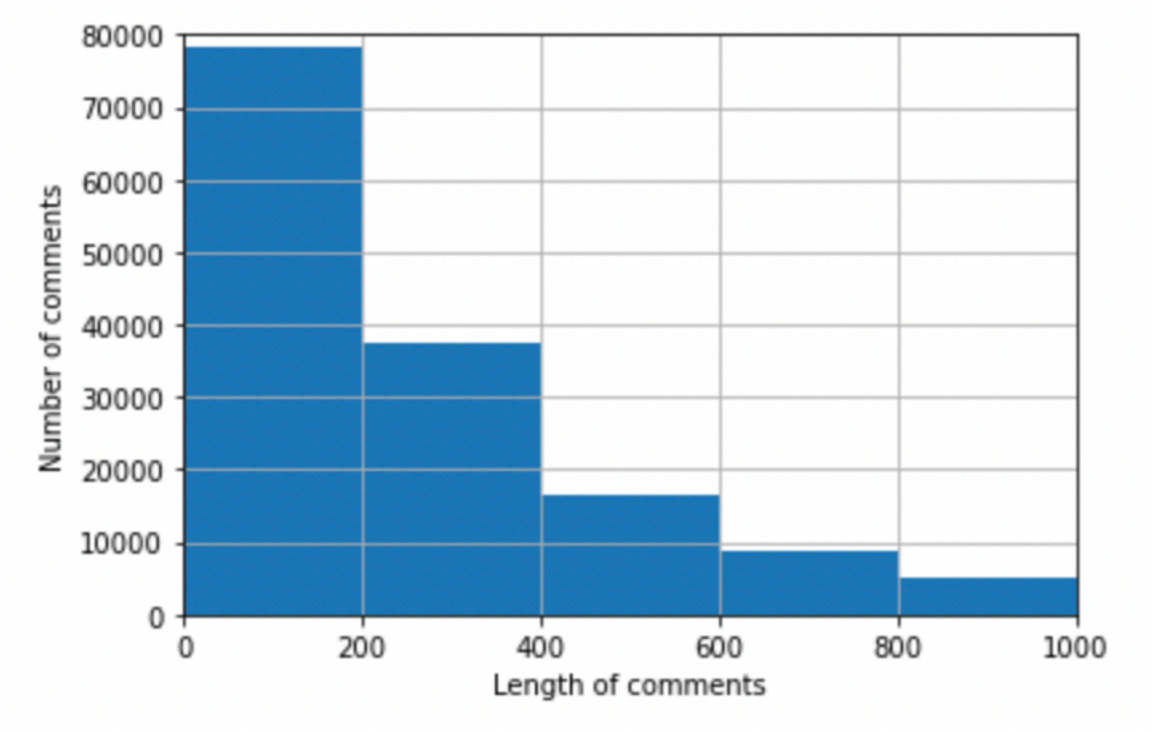
We can easily notice that approximately every 1 data out of 10 is malignant, every 1 in 20 samples is rude and abuse, but the occurrences of sample being highly_malignant, threat and loathe is extremely rare i.e., less than 1%.

Multi Labelling Level	Frequency	Percentage
One	6360	3.99
Two	3480	2.18
Three	4209	2.64
Four	1760	1.10
Five	385	0.24
Six	31	0.02

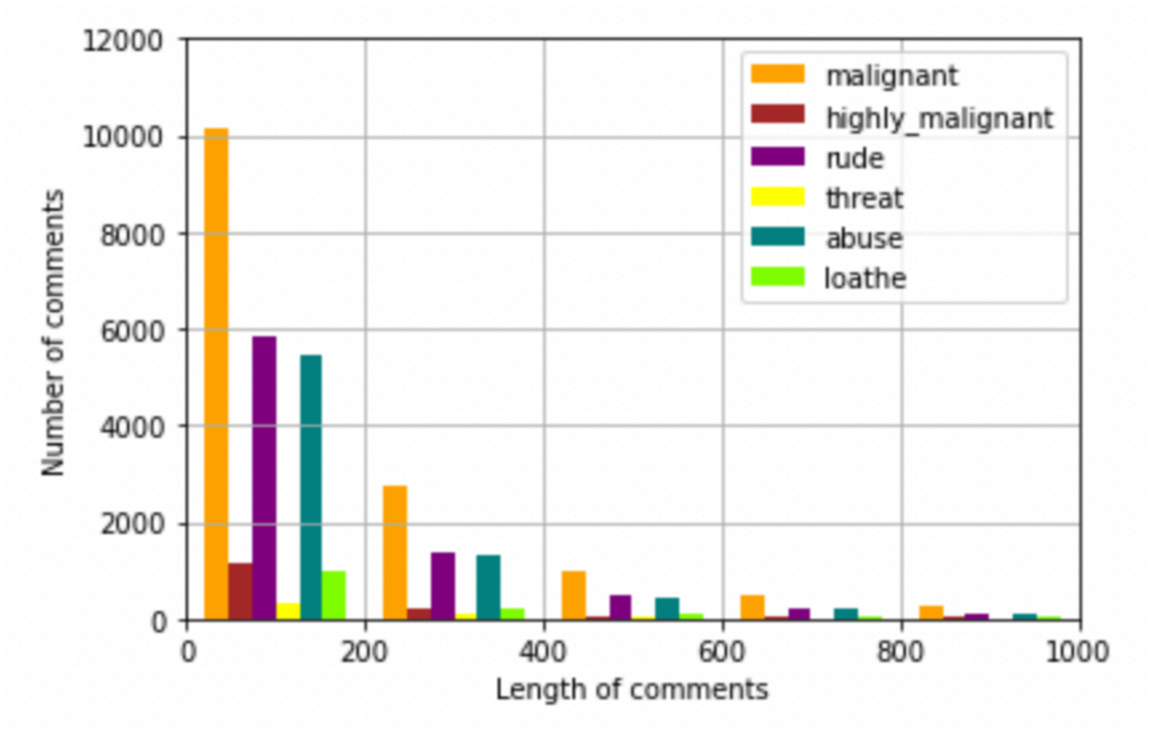
Overall 6360 samples are those which have at least one label, 3480 samples have 2, 4209 have 3, 1760 have 4, 385 have 5 and only 31 have six labels.

4. Exploratory Visualisation

- From the first visualisation we can observe that comments have varying lengths from within 5 up to 5000 words. The majority of comments have length up to 400, and as we move towards greater lengths, the number of comments keep on falling. Since including very long length comments for training will increase the number of words manifold, it is important to set a threshold value for optimum results.



- From the second visualisation, we can observe the number of words falling under the six different outcome labels toxic, severe_toxic, obscene, threat, insult and identity_hate along with their lengths.



- Here also similar to the first plot, we observe that most of the malignant, rude and abusive comments have lengths under 200, and this number falls with the length of comments.

- Based on these plots, taking comments having lengths upto 200 for training is a good estimation of our data and can be expected to give acceptable results on testing later. Hence before preprocessing, we will be removing all comments with length more than 200 which will serve as our threshold.

5. Algorithms & Techniques

As discussed in the Problem Statement section, any multi-label classification can be solved using either Problem Statement methods or Adaptation Algorithms.

Problem Transformation Methods:

- **Binary Relevance Method**: This method does not take into account the interdependence of labels. Each label is solved separately like a single label classification problem. This is the simplest approach to be applied.

- **Classifier Chain Method**: In this method, the first classifier is trained on input data and then each of the next classifier is trained on the input space and previous classifier, and so on. Hence this method takes into account some interdependence between labels and input data. Some classifiers may show dependence such as toxic and severe_toxic. Hence it is a fair deal to use this method.

- **Label Powerset Method**: In this method, we consider all unique combinations of labels possible. Any one particular combination hence serves as a label, converting our multi-label problem to a multi class classification problem. Considering our dataset, many comments are such that they have 0 for false labels all together and many are such that obscene and insult are true together. Hence, this algorithm seems to be a good method to be applied.

Adaptation Algorithms:

- **MLKNN**: This is the adapted multi label version of K-nearest neighbors. Similar to this classification algorithm is the BRkNNaClassifier and BRkNNvClassifier which are based on K-Nearest Neighbors Method. This algorithm proves to give superior performance in some datasets such as yearst gene functional analysis, natural scene classification and automatic web page categorization.

Since this is somewhat similar to the page categorization problem, it is expected to give acceptable results. However, the time complexity involved is large and therefore it will be preferable to train it on smaller dataset.

Analytical Problem Framing

1. Implementation

- We will be defining **function evaluate_score** for printing all evaluation metrics: Some implementations return a sparse matrix for the predictions and others return a dense matrix. So, a try except block were used to handle it.

- Then, we will start implementing various **problem transformation methods**. Binary Relevance, Label Powerset and Classifier Chain methods were included.
- **Binary Relevance** method were implemented from scratch. It does not consider the interdependence of labels and basically creates a separate classifier for each of the labels.
- Next, Binary Relevance method for other classifiers (SVC, multinomialNB, gausseanNB) is directly imported from the **Scikit-multilearn** library. Classifiers for **Classifier Chain** and **Label Powerset** are imported and tested.
- Then **Adaptation Algorithms** were used. To be precise, MLkNN will be imported from the scikit-multilearn library and BP-MLL will be implemented from scratch.
- The **evaluate_score function** cannot be used to operate directly on the predictions of this model as it returns **probabilities in a range of values from 0 to 1**. The hamming_loss and accuracy_score cannot work on these values directly. Hence, predicted results were rounded.

2. Refinement

Problem Transformation Methods:

- Our initial model was the Binary Relevance method using **MultinomialNB** classifier.
- We then tried other classifiers like **SVC** and **GaussianNB**:
- Since MultinomialNB was giving the best results until now, **Classifier Chain** and **Label Powerset** was implemented using this classifier only.

Adaptation Algorithms:
MLkNN was implemented to get further improved results.

3. Results

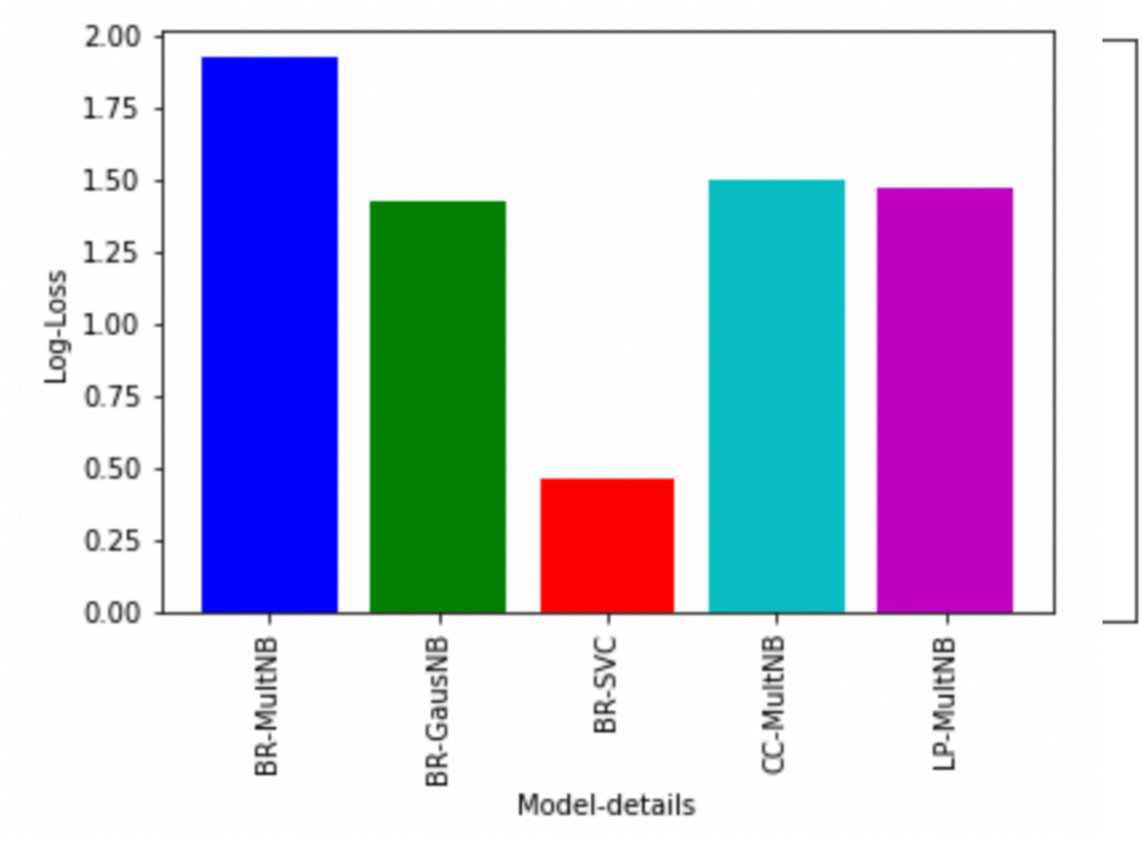
We can compare our results with the SVM model which we generated during the implementation section while we were trying out different classifiers with the Binary Relevance Method.

Free Form Visualisation
The hamming-loss and log-loss of different models used. We plotted in 2 scatter plots:

- If we compare all the models on hamming-loss: The best model will be LP-MultiNB i.e. Label Powerset Model with Multinomial NB Classifier.

Model	Hamming Loss	Log Loss	Accuracy
BR with Multinomial NB Classifier	4.21	2.82	84.89
BR with SVM Classifier	2.94	2.48	87.89
BR with Multinomial Classifier	4.21	2.82	84.89
BR with Gaussian NB Classifier	23.85	1.98	45.50
Classifier Chain with Multinomial NB	4.45	2.18	85.29
Label Powerset with Multinomial NB Classifier	3.87	1.91	85.89
ML kNN	5.11	2.01	82.19

- If we compare all the models on log-loss: The best model will be BR with SVM Classifier.



Thus, BR with SVM Classifier is selected as our final model

Conclusion

- The first step involved collecting data and deciding what part of it is suitable for training. This step was extremely crucial since including only very small length comments would give poor results if the length was increased whereas including very long length comments would increase the number of words drastically, hence increasing the training time exponentially and causing system (jupyter kernel) to go out of memory and die eventually.

- The second major step was performing cleaning of data including punctuation removal, stop word removal, stemming and lemmatizing. This step was also crucial since the occurrence of similar origin words but having different spellings will intend to give similar classification, but computer cannot recognize this on its own. Hence, this step helped to a large extent in both removing and modifying existing words.

- The third step was choosing models to train on. Since I had a wide variety of models(3 for problem transformation) and classifiers(not bounded) along with number of adaptation models in MLL kNN, selecting which all models to train and test took lots of efforts.

- Finally comparing on the basis of different evaluation metrics. The two major evaluation metrics I planned to compare on were hamming-loss and log-loss. Hence the final model selection was done on the basis of the combination of both these losses.