Malignant Comment Classifier

Seep Bansal | Machine learning | 12-03-2010

ACKNOWLEDGEMENT

I would like to express my special thanks of gratitude to the team Data Trained and my Mentor MS Deepika Sharma for their exemplary guidance, monitoring and constant encouragement thought the journey of learning Data science and Machine learning techniques. I would also like to express my heartly gratitude to the support team of data trained for their constant support. Last but not the least, I would also like to thank the team of Flip Robo technologies for giving me this opportunity to work on this project and the mentors in Flip Robo Technologies who are constantly guiding me to enhance my knowledge and work. This project helped me not only to learn how to do proper research but also helped me in learning many new things.

Problem Definition

Project Overview Online platforms when used by normal people can only be comfortably used by them only when they feel that they can express themselves freely and without any reluctance. If they come across any kind of a malignant or toxic type of a reply which can also be a threat or an insult or any kind of harassment which makes them uncomfortable, they might defer to use the social media platform in future. Thus, it becomes extremely essential for any organization or community to have an automated system which can efficiently identify and keep a track of all such comments and thus take any respective action for it, such as reporting or blocking the same to prevent any such kind of issues in the future. This is a huge concern as in this world, there are 7.7 billion people, and, out of these 7.7 billion, more than 3.5 billion people use some or the other form of online social media. Which means that every one-in-three 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 toxicity like threats, obscenity, insults, and identity-based 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.

Problem Statement

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.

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.

Our goal is to 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.

Analysis

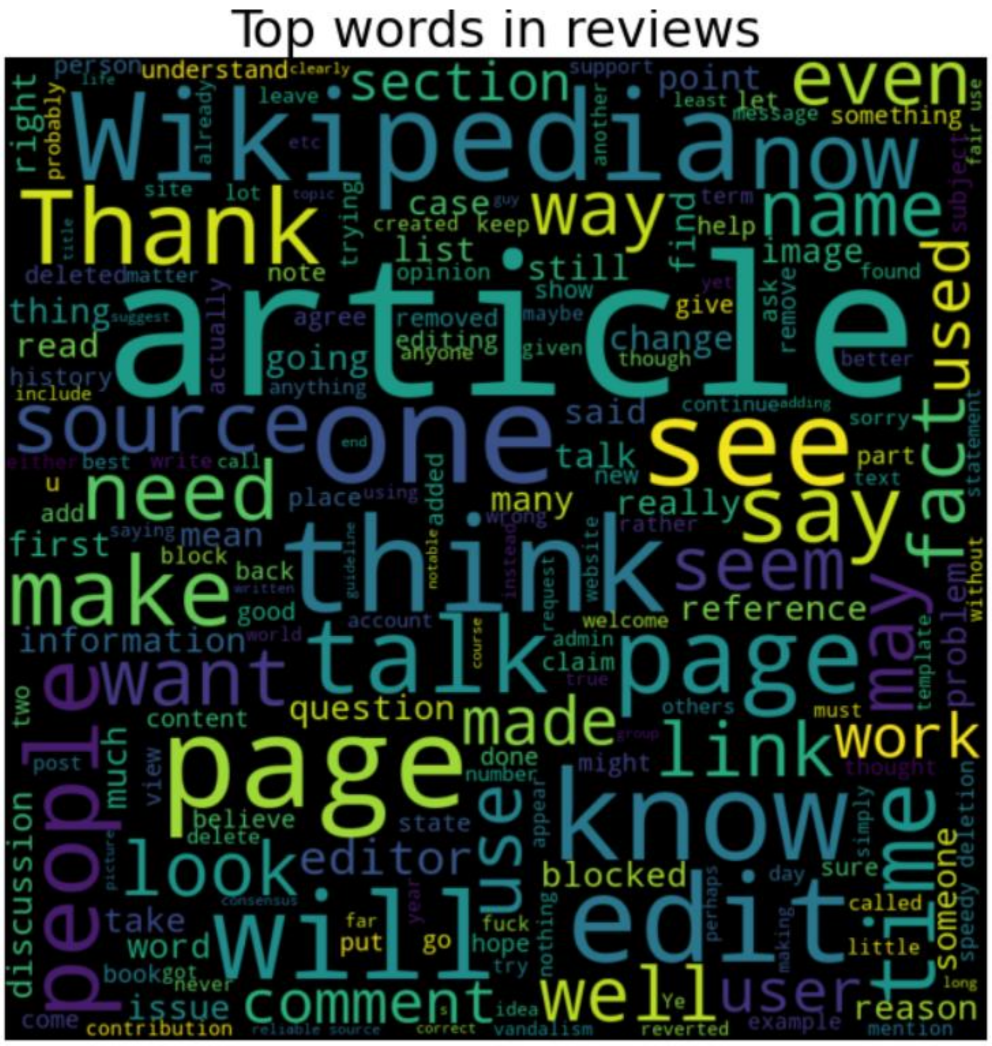
The data set contains the training set, which has approximately 1,59,000 samples and the test set which contains nearly 1,53,000 samples. All the data samples contain 8 fields which includes 'Id', 'Comments', 'Malignant', 'Highly malignant', 'Rude', 'Threat', 'Abuse' and 'Loathe'.

The label can be either 0 or 1, where 0 denotes a NO while 1 denotes a YES. There are various comments which have multiple labels. The first attribute is a unique ID associated with each comment.

The data set includes:

- **Malignant:** It is the Label column, which includes values 0 and 1, denoting if the comment is malignant or not.
- **Highly Malignant:** It denotes comments that are highly malignant and hurtful.
- **Rude:** It denotes comments that are very rude and offensive.
- **Threat:** It contains indication of the comments that are giving any threat to someone.
- **Abuse:** It is for comments that are abusive in nature.
- **Loathe:** It describes the comments which are hateful and loathing in nature.
- **ID:** It includes unique Ids associated with each comment text given.
- **Comment text:** This column contains the comments extracted from various social media platforms.

Data Visualization



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.

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.

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. 7 In this, we find that x1 and x4 have the same labels, similarly, x3 and x6 have the same set of labels. So, label power set transforms this problem into a single multi-class problem.

ADAPTION 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.

BP-MLL Neural Networks: Back propogation Multi-label Neural Networks is an architecture that aims at minimizing pair-wise ranking error. An architecture of one hidden layer feed forward neural network is as follows: 8 The input layer is fully connected with the hidden layer and the hidden layer is fully connected with the output layer. Since we have 6 output labels, the output layer will have 6 nodes. This algorithm can be trained in a reasonable amount of time with appropriate number of nodes in the hidden layer.

4. Benchmark Model As we proposed in our proposal, we will be using Support Vector Machine with radial basis kernel (rbf) as the benchmark model (using Binary Relevance Method) Implementation of this model has been done along with other models using the Binary Relevance Method in the implementation section. Since we have a large dataset, other classifiers using the bag of words model such as the MultinomialNB and GausseanNB are expected to work than this model. But, in practice, it performs quite well. A detailed comparison and analysis between all these classifiers will hence be provided

3. Methodology

Data Preprocessing The following steps were taken to process the data: → A string without all punctuations to be prepared: 1. The string library contains punctuation characters. This is imported and all numbers are appended to this string. Our comment_text field contains strings such as won't, didn't, etc. which contain apostrophe character('). To prevent these words from being converted to wont or didn't, the character ' represented as \' in escape sequence notation is replaced by empty character in the punctuation string. 2. make_trans(intab, outtab) function is used. It returns a translation table that maps each character in the intab into the character at the same position in the outtab string. → Updating the list of stop words: 1. Stop words are those words that are frequently used in both written and verbal communication and thereby do not have either a positive or negative impact on our statement like "is, this, us, etc.". 2. Single letter words if existing or created due to any preprocessing step do not convey any useful meaning and so they can be directly removed. Hence letters from b to z, will be added to the list of stop words imported directly.

Stemming and Lemmatizing: 1. The process of converting inflected/derived words to their word stem or the root form is called stemming. Many similar origin words are converted to the same word e.g. words like "stems", "stemmer", "stemming", "stemmed" as based on "stem". 2. Lemmatizing is the process of grouping together the inflected forms of a word so they can be analyzed as a single item. This is quite similar to stemming in its working but differs since it depends on correctly identifying the intended part of speech and meaning of a word in a sentence, as well as within the larger context surrounding that sentence, such as neighboring sentences or even an entire document. 10

3. The wordnet library in nltk will be used for this purpose. Stemmer and Lemmatizer are also imported from nltk. → Applying Count Vectorizer:

To convert a string of words into a matrix of words with column headers represented by words and their values signifying the frequency of occurrence of the word Count Vectorizer is used.

Stop words were accepted, convert to lowercase, and regular expression as its parameters. Here, we will be supplying our custom list of stop words created earlier and using lowercase option. Regular expression will have its default value. → Splitting dataset into Training and Testing:

Since the system was going out of memory using train_test_split, I had jumbled all the indexes in the beginning itself.

The shuffle function defined here performs the task of assigning first 2/3rd values to train and remaining 1/3rd values to the test set.

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. 11 → 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. The code is as follows: → 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. 12 → 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. → Back Propagation MultiLabel Neural Networks (BP-MLL) can be implemented using the Sequential Model from Keras. Checkpointer is used to show the intermediate results from different epochs. → The model architecture is as follows: → 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. Justification 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. The SVM model had hamming loss of 4.267% while the log loss was 0.461. Comparing with our best model i.e – → MultinomialNB model in LP method (hamming loss of 3.17% and log loss of 1.47) - Hamming loss improved by 1.09% - Log loss increased by 1.009 → BP-MLL model with params as {nodes in hidden layer = 16, learning rate = 0.001, epochs = 10, batch size = 64} and (hamming loss of 15.15 and log loss of 0.35) - Hamming loss increased by 10.8% - Log loss decreased by 0.11 Thus, our model has some improvement over the SVM model