AskReddit Troll Question Detection Challenge

Team: Gduo_UltraProMax

Ayush Das
IMT2019014

Balkaran Singh
IMT2019016

Abstract: **- Reddit is a less-well-known social media website built on anonymity and somewhat democratic principles. The probability of your post/comment "going viral" on Reddit is the same irrespective of whether you are a celebrity or if you're a commoner. Each community or "subreddit" has its own topics and rules enforced by subreddit moderators. The admins of Reddit only step in when there's a public controversy or when their Terms of Service are violated.**

AskReddit is one of the most popular subreddits. Their official description: r/AskReddit is the place to ask and answer thought-provoking questions.

Of course, not all questions here are thought-provoking. Some are very obvious "troll" questions.

The moderators of AskReddit continuously strive to remove such questions, however, it is not possible to manually inspect every question and do this -- thousands of questions are added every day.

Which is where YOU come in. The moderators of AskReddit have provided you with a sample of all the questions they received last year. Your job: To create a model capable of automatically detecting troll questions so that they can be flagged and removed.

1. Overview

Reddit is an American social news aggregation, web content rating, and discussion website. The website is based on the concept of complete anonymity. Spam posts are a big problem on Reddit. Popular social media websites like Twitter and Facebook hire people who decide whether the post is spam or non-spam whereas moderation on Reddit is conducted by community-specific moderators, who are not Reddit employees. Since Reddit is completely anonymous, it is flooded with spam questions and it is not possible for moderators to manually mark a question spam or non-spam.

In this paper, we have tried to solve this problem. We have used the dataset provided and built some machine learning models to solve this problem. Our goal was, given a question, our model should be able to classify the question as either spam or non-spam. For example, “How do I pretend to be a liberal so that I can get into college?” is a spam question whereas “Why do elevators go super slow right before the doors open?” is a valid question.

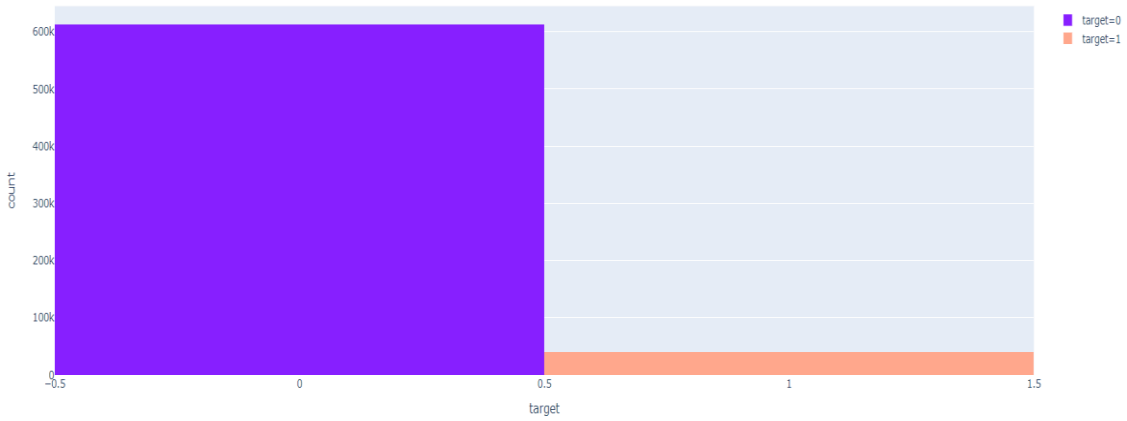
2. Dataset

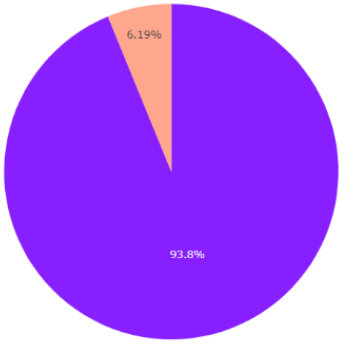
Train dataset provided for the Kaggle competition has 6,53,061 rows and 3 columns (question_id, question_text, target). The ‘question_text’ column contains the question and the ‘target’ column respectively contains the target value which is binary i.e., it takes values 0 or 1.
Test dataset has 6,53,061 rows and 2 columns (question_id, question_text). The ‘question_text’ column contains the question for which we have to predict the binary labels.

3. Exploratory Data Analysis (EDA)

Before doing pre-processing, we performed EDA on our dataset which helped us understand it better.

a.) Count of number of zeroes and ones.

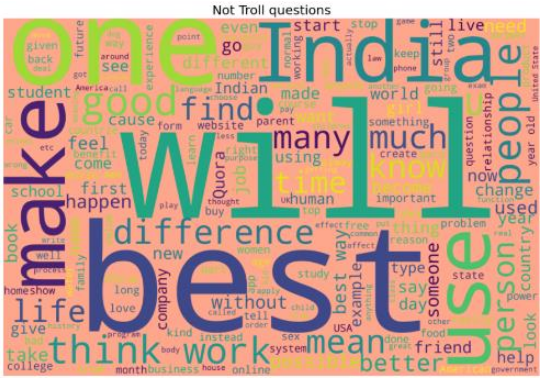




From the above two plots, it can be deduced that the number of zeroes (non-spam questions) are far more than the number of ones (spam questions) in our train dataset. From this we can say that our train dataset is biased/ skewed.

b.) Wordclouds for spam questions and non-spam questions.

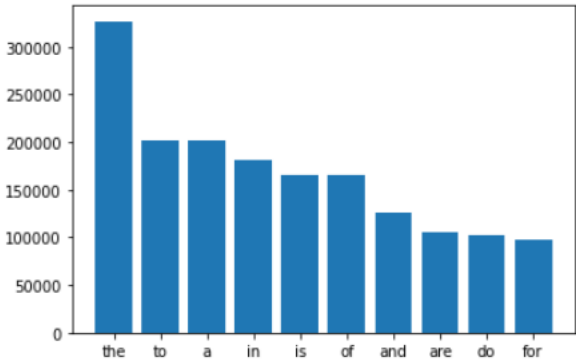
Wordclouds are visual representation of words that give greater prominence to words that appear frequently. Size of words in Wordclouds depends on the frequency with which they appear.



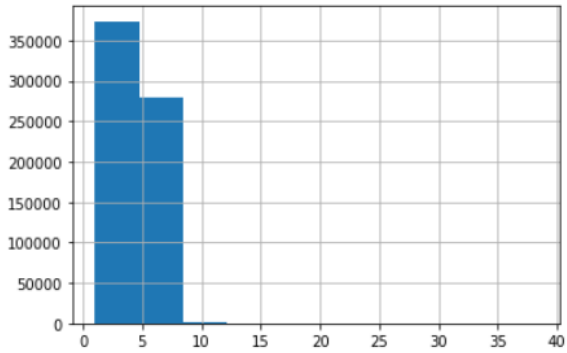
The above word cloud is for non-spam questions. From the above plot we deduced that common words like will, make, best, one etc. are more frequent in non-spam questions.

The above word cloud is for troll questions. From the above plot we deduced that words like people, women, India, Trump etc. are more frequent in spam-questions.

c.) Plot for top stop words. Stopwords are a set of commonly used words in a language. In English language some stopwords are “a”,” the”,” is”,” are”, etc.

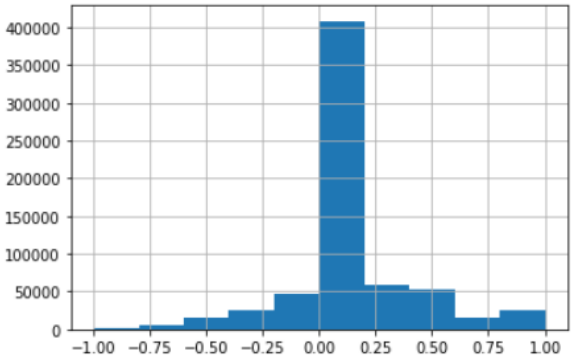


d.) Histogram for word length.



From the above plot it can deduced that more than 99% words have length less than or equal to 5 characters.

e.) Plot of polarity (Sentiment analysis).



The above plot gives a sentimental analysis of the questions. It is made using Textblob. Textblob is a python library built on top of nltk. The sentiment function of Textblob returns,

i) Polarity: - is a floating-point number that lies in the range [-1,1] where 1 means positive statement and -1 means a negative statement.

ii) Subjectivity: - is a floating-point number that lies in the range [0,1]. It refers to how someone's judgement is shaped by personal opinions and feelings.

In the above plot, the x axis shows the polarity of the question and the y axis shows the number of questions. It can be deduced from the plot that most of the questions have a neutral polarity (polarity = 0) i.e., they are neither too negative nor too positive.

4. Data Cleaning and Pre-processing

Data cleaning refers to identifying and correcting errors in the dataset that may negatively impact a predictive model. Data pre-processing refers to the technique of preparing the raw data to make it suitable for building and training machine learning models. Data pre-processing is a very important step in any machine learning problem. The extent of pre-processing and cleaning plays an important role in determining the final score of machine learning problem.

a.) Cleaning data

The first thing that we did was cleaning of our train dataset. We made a function 'clean_text' which takes in as input a text statement and does the following to it,

i) *Removes all the numbers,*

ii) *Removes all the punctuation marks,*

iii) *Removes all the special characters (@, #, $, %, etc.),*

iv) *Removes all the extra spaces*,

v) *Removes all the stopwords,*

vi) *Converts all the alphabets to lowercase alphabets.*

For example, the question "What happened in revolt of 1857?" after cleaning becomes "happened revolt". The question essentially looses its meaning but only important words are left after cleaning.

b.) Stemming and Lemmatization

Stemming and Lemmatization are two important text normalization techniques in Natural Language Processing.

Stemming is the process of producing morphological variants of a root/base word. Stemming programs are commonly referred to as stemming algorithms or stemmers. A stemming algorithm reduces the words "chocolates", "chocolatey",

"choco" to the root word, "chocolate" and "retrieval", "retrieved", "retrieves" reduce to the stem "retrieve".

Lemmatization is the process of grouping together the different inflected forms of a word so they can be analysed as a single item. Lemmatization is similar to stemming but it brings context to the words. So, it links words with similar meanings to one word.

Both Stemming and Lemmatization have their own pros and cons. We implemented both of these and saw a better F1 score with stemming.

There are a plenty of stemming algorithms available but we decided to go with **Snowball Stemmer.**

```
program  :  program
programs  :  program
programmer  :  program
programming  :  program
programmers  :  program
```

There are mainly two errors in stemming: -

i) Over-stemming: - It occurs when two words are stemmed from the same root that are of different stems. Over-stemming can also be regarded as false positives.

ii) Under- stemming: - It occurs when two words are stemmed from the same root that are not of same different stems. Under-stemming can also be regarded as false positives.

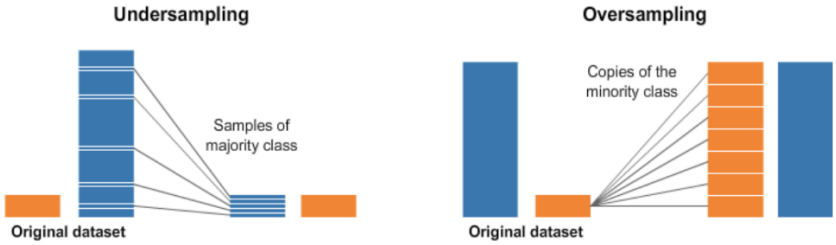
We used Snowball Stemmer as it can map non-English words too. It is way more aggressive than Porter Stemmer and is also the most widely used stemmer.

c.) Resampling

While doing EDA we estimated the number of ones and the number of zeroes. There are over 6 lakh zeros and merely 40 thousand ones. This means our dataset is skewed. If we train on this dataset then we won't get a good accuracy as the representation of ones is way lees than the representation of zeroes. To counter this, we use resampling. There are essentially two types of resampling techniques namely undersampling and oversampling.

i) Undersampling: - In undersampling we randomly delete examples in the majority class i.e., we decrease the number of zeroes in our case.

ii) Oversampling: - In oversampling we randomly duplicate examples in the minority class i.e., we increase the number of ones in our case.



We tried both undersampling and oversampling but we got best F1 score using undersampling.

d.) CountVectorizer

CountVectorizer is a great tool provided by the scikit-learn library in Python. It is used to transform a given text into a vector on the basis of the frequency (count) of each word that occurs in the entire text. CountVectorizer creates a matrix in which each unique word is represented by a column of the matrix, and each text sample from the document is a row in the matrix. The value of each cell is nothing but the count of the word in that particular text sample.

We used both tfidf vectorizer and CountVectorizer but we were getting best F1 score with CountVectorizer.

5. Model Selection

Model selection is an important and difficult part in any machine learning problem. We tried plenty of models and calculated their respective accuracy on train dataset. Mainly we had used LogisticRegression, Naïve Bayes classifier, Random Forest classifier, Xgboost, Adaboost, Stochastic GD Classifier and SVC.

Model	Accuracy on train dataset	F1 score on test dataset
Logistic Regression	0.9357	0.61861
Naïve Bayes Classifier	0.9251	0.56843
Random Forest Classifier	0.9784	0.5428
Xgboost	0.8931	0.5746
Adaboost	0.9154	0.6012
SGD classifier	0.8654	0.5276

SVC	0.8779	0.5108
Stacking Classifier*	0.9473	0.5994

*Stacking Classifier using Random Forest Classifier, SVC, and logistic regression

Since we got best F1 score on test dataset using Logistic Regression, we used it as our final model. To calculate accuracy on train dataset, we have split the train dataset in test and train using sklearn's train_test_split where we used 80% as train dataset and 20% as test dataset.

6. Conclusion

We tried different approaches, different algorithms and different models and concluded the contest with an accuracy of 0.61861. This project has a huge importance in real life applications and it becomes very important to tackle issues like these.