Quora Insincere Question Detection

Shweta Masrani
MT2020051
International Institute of Information Technology, Bangalore

Akshil Gadhiya
MT2020007
International Institute of Information Technology, Bangalore

Alay Dhagia
MT2020102
International Institute of Information Technology, Bangalore

Abstract - The Internet contains an endless supply of knowledge and information that allows to learn about almost any topic or question. Apart from Google, there are websites where you can ask any question and people around the world answer it specifically. Quora is one such website for question answers where one can ask questions from any domain and others answer to them. But like other social media sites, this is also misused by asking inappropriate questions. A question tagged as an 'insincere' question if it includes a non neutral tone, is disparaging, inflammatory or the questions are in negative sexual aspects and if it is not grounded to reality. In this project we are trying to detect insincere questions using traditional machine learning algorithms.

Index Terms — Feature Engineering, XGBoost, Grid Search, Cross Validation, Logistic Regression, Naive Bayes, Random Forest Classifier, SGD Classifier

I. Introduction

In the growing era of social media almost each and everyone has used them in some way or other. Quora is one such social media network which serves as a questions and answer platform where users ask the questions and other users answer them. On such a platform where almost the entire data is generated by users it is axiomatic that some insincere data will also be present. To keep the content over the website clean some sort of regulation has to be applied over it. Quora as a platform needs to make sure that the content over the website is sincere and does not affect sentiments, values and happiness of the users and along with that prevent users from leaving the platform because of such content.

This paper is going to tell the steps taken in our project in order to build a machine learning model which will help predicting insincere questions from a lot of data containing the questions.

Depending on the results given by the model Quora can delete the question or ban the user or take any other appropriate decision.

II. Dataset

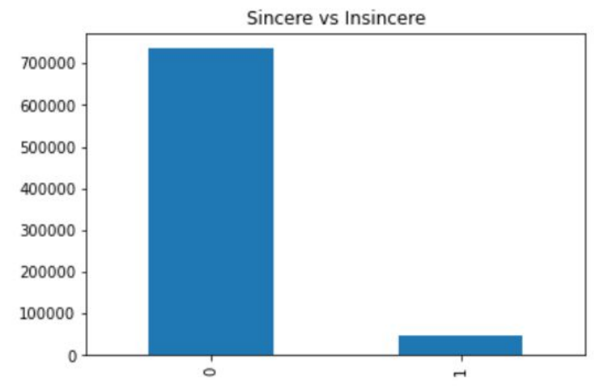
The dataset we are using here is available at the Quora Insincere Questions Classification competition hosted on kaggle. The data contains questions collected from Quora web portal. Each data point in training data consists of 3 columns : qid, question_text (original question) and target (0 - sincere question, 1 - insincere question). The testing data is not labelled. There are 7,83,673 data points available in training data and 5,22,449 data points available in testing data.

Here, it can be inferred that the problem is of binary classification.

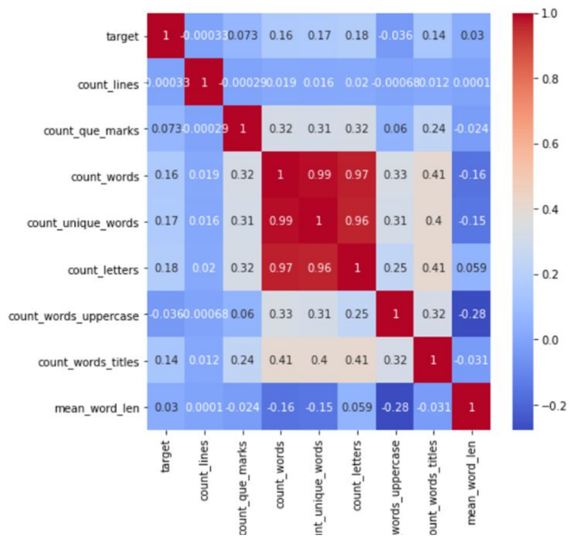
III. Observation

Before going ahead with preprocessing it is good to have some insights about the given dataset.

The very first thing we observed is that the class distribution of sincere vs insincere questions is highly imbalanced. Over 93% of the questions are sincere and less than 7% of the questions are actually insincere.



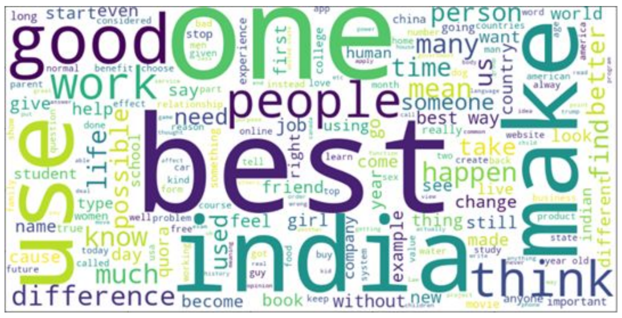
It is a common trend on the internet nowadays to write in all upper case when the writer wants to convey anger or hate. With an assumption that the length of the question and the way of writing the question might have some correlation with the target variable, we plotted the correlation matrix using different components of sentence structure. We added features like length of question, upper case words, use of unique words and question marks etc.



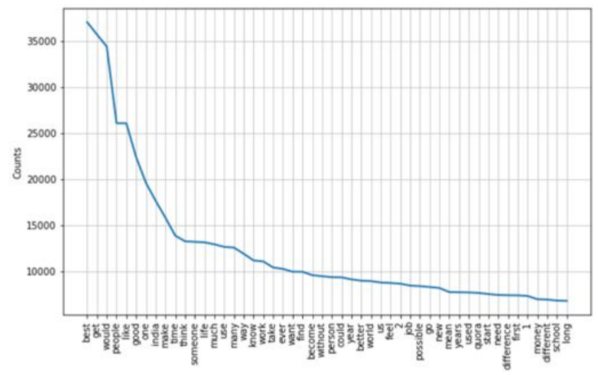
The added features do not show strong correlation as such. But, the count_words and count_unique_words shows strong correlation that implies there are very less repetitive words in questions.

Further, to know what kind of words are there in each target class we used word cloud and word frequency graphs.

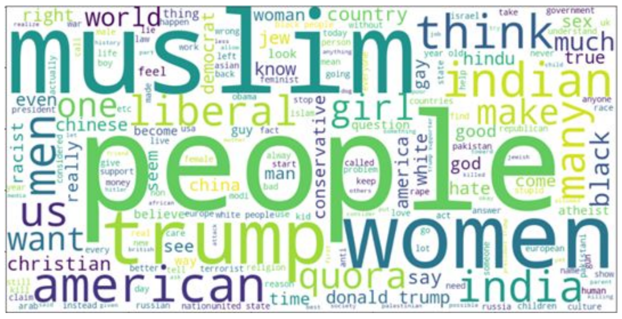
Sincere class word



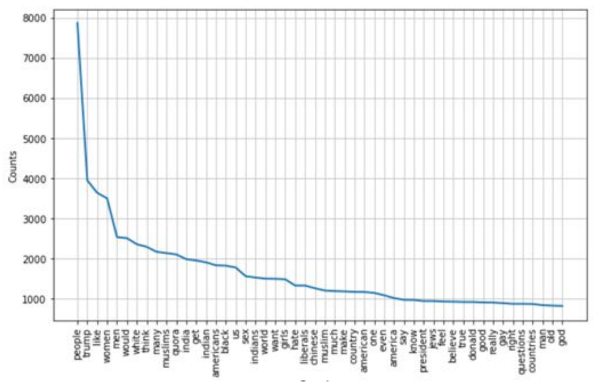
Sincere class word frequency graph



Insincere class word cloud



Insincere class word frequency graph



It is clear from the word cloud and the frequency graphs that there are some words which are common to both the classes and the frequency of some of these common words are also pretty high in both the classes respectively i.e. the word ‘people’ is highest occurring word in insincere class and forth highest occurring in sincere class. The most commonly occurring words which are the same in both the classes would not add much value to training so we are removing these words.

IV. Data Preprocessing and Feature Extraction

Text preprocessing is traditionally an important step for natural language processing (NLP) tasks. It transforms text into a more digestible form so that machine learning algorithms can perform better.

For data cleaning and preprocessing steps we used libraries like nltk, regex, re, and autocorrect for spelling correction. We removed all the numbers and hyperlinks from the questions because that is not helpful to decide the insincerity of a question. Further, we removed contractions from the words like “don’t” become “do not” and “I’ve” becomes “I have”. We also removed punctuations and white spaces from the questions.There are some questions in the corpus that contain the special symbols and letters or words from other languages than english i.e. chinese. We also removed those. There are questions with incorrectly spelt words like “awwards” which is “awards”. We used the autocorrect library to correct spelling. We used lemmatization and removed stop words from the questions but at a later stage we realized that by these steps we are actually losing out on context and some information which might be helpful to train the model better so we trained the model without these two steps which gave better results.

Further, to handle imbalance distribution of target class we used various methods of sampling namely random under sampling and SMOTE oversampling. In random undersampling the data points from majority class are randomly deleted to balance both the classes. Here there is a chance that we might lose out on important information. Hence we used SMOTE - Synthetic Minority Oversampling Technique which is used to synthesize new data points of minority class using existing data points.SMOTE uses KNN (K- nearest neighbour) algorithm to synthesize new data points, where it does not differ between another class’s data points hence decreasing the data variability and it introduces correlation between samples and resulting in blurred decision boundary. We continued to word with imbalance data after trying out these techniques.

We started with two simple approaches, BagOfWords and feature hashing. BagOfWords creates a vocab extracting words and keeping the frequency count. There are many downfalls of this method, few of them are that it does not maintain the rareness and weightage of the words. Feature Hashing hashes the question data in order to make features. Very quick and easy to implement but not much helpful in training data and also once hashed we cannot get the question back which is a big drawback.

We tried different word embeddings like google news, Glove, Word2Vec etc. One major drawback of word embeddings is its inability to handle unknown words or Out Of Vocabulary words.

Finally, to extract the features from the questions we used TF-IDF vectorizer (Term Frequency - Inverse Document Frequency). TF-IDF gives us a way to associate each word in a document with a number that represents how relevant each word is in that document.

1. `Term Frequency = (Number of time the word occurs in the text) / (Total number of words in text)`
2. `Inverse Document Frequency = (Total number of documents / Number of documents with word t in it)`
3. `TF-IDF = TF * IDF`

We used TF-IDF over the combined text of train and test in order to capture all the words occurring in the data with maximum features as 40,000 for an n gram of (1,1). In addition to using TF-IDF vectorizer for the words we also employed it on character n-grams of these words (2-4 including spaces) with maximum features of 30,000. Then stacked both these matrices using scipy’s hstack. Additionally we used the l2 norm while calculating TF-IDF to normalize the vectors. Normalizing lets us consider the frequency of words relative to each other, while removing the effect of total word count.

V. Model Selection

After performing various types of preprocessing and feature extraction techniques we applied a brute force combination of all the models as and when we learned in the class and understanding if they can be helpful in our project. We went through and tried various classification models and ensemble models and used them with and without parameter tuning and checked the results. Logistic regression, SVM, naïve bayes, decision tree, random forest, SGD, XGBoost, Stacking and LightBGM are the models which were applied by us. Following are the observations and results obtained from each.

Logistic regression: This is a simple model but immensely powerful model[8]. By performing various changes to this model and the data given to it this model outperformed all the other models we used. Initially we used TFIDF on the actual training preprocessed data with word vectorizer. We did not tune the parameters and applied the model with default parameters, but the results were not significant enough. Also, the model used to run out of memory, so we had to limit the max features used in TFIDF. Various combinations of max features were also tried. Since there was a high imbalance between the sincere and insincere data, we used the model on under-sampled and oversampled data. Score obtained from these modifications ranged somewhere between 0.54-0.60

All the above variations were also tried with TF-IDF vectorizer with n-gram range but again the results were not good enough. Good combination of trials for TF-IDF with word, char vectorizer(n-gram range 2-4) and under-sampled and oversampled data were tried.

We also used a not-so-very practical approach to improve our score. Using predict_proba () on the model trained with the above combinations of different approaches on data we looped over the values 0.01 to 0.99 and tried to find an optimal threshold, to remove positives which had probability less than the threshold. Default threshold is 0.5 which was changed.

By this approach, we could reach a score of 0.601 which was the highest till what we had tried till that point of time.

Since balancing the data was not improving the score, we dropped the idea of using it.

Parameter tuning which could help:

C: values tried {0.01, 0.1, 1, 10, 100}. Best score with 1

Solver: {liblinear, saga, lbfgs}. Best score with lbfgs optimization algorithm

Doing this without changing the threshold best score obtained was 0.58

After trying various models and using various approaches we observed that if we do not perform stopwords removal and lemmatization it helped training the model more properly.

Model without parameter tuning gave a score of 0.62 while with parameter tuning gave us a score of 0.63 which was the best we could get of everything.

Apart from TF-IDF we also used BagOfWords, Feature Hashing and Word2vec over the models but TF-IDF was clearly the winner from all the approaches taken.

All the following models are tried with the different combinations stated over TF-IDF.

SVM: This model was also tried with all the combinations stated above along with all the possible parameter tunings we could do. The scores were not appreciating, and it took a long time to converge as well.

Naïve bayes: We took this model into consideration because of its simplicity and given how quick it is in training. Since this model takes into consideration that no features are correlated to each other the outputs would tend to become more probabilistic.

Best score: 0.54

Decision tree: We tried this model without specifying the dept of the tree and let the model itself come up with a depth between 1 to 32. However, this model took a long time to train hence we tried to manually try many values of depth. Other parameter tunings were also applied like changing the criterion which measures the quality of split. This model overfit the data.

Score obtained: 0.43

After trying the above models, we moved to ensemble method [9].

XGBoost: We applied it without any parameter tuning but it took a long time and eventually did not converge. Tuned the parameter early_stopping_rounds still the model did not trained. Finally changing the preprocessed data, the model did converge.

LightGBM: Applied before any stopwords removal and lemmatization was removed.
Lgbm traverses in DFS manner. Advantage of is that it does not consume much memory.
Parameters tuning:
n_estimators: It defines the number of boosted trees to use. Default value is 100 but for this model we got best results using relatively higher n_estimators and lower learning_rate.
Best score for this model: 0.56

Stacking: 3 models trained above Logistic Regression, LightGBM, Decision Tree were stacked for use. Max voting parameter tune: 3 models above (0.605)
Score: 0.61
Noteworthy models and corresponding best scores:

Model	**Score**
Logistic regression	0.63
Naïve Bayes	0.54
Decision tree	0.43
XGBOOST	0.51
LightGBM	0.56

VI. Training details

Since we had a large amount of training data to work with, we had to be careful about feature extraction as it might create a load on the model. We handled the data in different ways depending on the feature extraction techniques used.

We split the training data into 80:20 ratio while training the model.

VII. Conclusion

Given that we were limited to using classical machine learning models we believe that we have achieved fairly good model to detect the insincerity of the Quora questions but obviously many topics are yet to be explored and various topics where we could delve deeper

VIII. Future Scope

Given classical models we could work more on decision trees and other tree-based algorithms to train a better model. Besides classical models' neural networks can help.

IX. Acknowledgement

We would like to thank Professor G. Srinivas Raghvan and Professor Neelam Sinha for great insightful lectures and well-designed course content. We especially like to thank our TA Nikhil Sai for being a very friendly, supportive and helpful mentor.
We would also like to thank other TAs who actively helped during the entire semester course and also came up with an idea of Kaggle competition so that in the chase of a good rank on the leaderboard we were also improving ourselves.

X. References

[1] E. Wulczyn, N. Thain, and L. Dixon, “Ex machina: Personal attacks seen at scale,” CoRR, vol. abs/1610.08914, 2016. [Online]. Available: http://arxiv.org/abs/1610.08914
[2] J. Cheng, C. Danescu-Niculescu-Mizil, and J. Leskovec, “Antisocial behavior in online discussion communities,” CoRR, vol. abs/1504.00680, 2015. [Online]. Available: http://arxiv.org/abs/1504.00680
[3] Kaitlyn, “Calculating tf-idf with python,” 2017, [Online; posted 3-June-2017]. [Online]. Available: \url{https://methodi.ca/recipes/calculating-Tf-idf-python}
[4] A. S. Hendrik Jacob van Veen, Le Nguyen The Dat, “Kaggle ensembling guide,” 2015, [Online; posted 6-February-2018]. [Online]. Available: \url{https://mlwave.com/kaggle-ensembling-guide/}
[5] Upasana, “How to handle imbalanced classification problems in machine learning?” 2017, [Online; posted 7-March-2017].[Online]. Available: \url{https://www.analyticsvidhya.com/blog/2017/03/imbalanced-classification-problem/}

[6] C. Liu, Y. Sheng, Z. Wei, and Y. Yang, “Research of text classification based on improved tf-idf algorithm,” in 2018
IEEE International Conference of Intelligent Robotic and Control Engineering (IRCE), Aug 2018, pp. 218–222.
[7] S. T. Indra, L. Wikarsa, and R. Turang, “Using logistic regression method to classify tweets into the selected topics,” in 2016 International Conference on Advanced Computer Science and Information Systems (ICACSIS), Oct 2016, pp. 385–390.
[8] and, “A comparison of several ensemble methods for text categorization,” in IEEE International Conference onServices Computing, 2004. (SCC 2004). Proceedings. 2004, Sep. 2004, pp. 419–422.
[9] Y. Liu, J. Niu, Q. Zhao, J. Lv, and S. Ma, “A novel
text classification method for emergency event detection on social media,” in 2018 IEEE SmartWorld, Ubiquitous Intelligence Computing, Advanced Trusted Computing, Scalable Computing Communications, Cloud Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI),
Oct 2018, pp. 1106–1111.