Quora Insincere Questions Classification

Team: ML_Noobs

Anish Rajan
IMT2018009
IIIT Bangalore
Bangalore, India
anish.rajan@iiitb.org

Gaurav Goel
IMT2018025
IIIT Bangalore
Bangalore, India
gaurav.goel@iiitb.org

Ishmeet Singh Saggu
IMT2018030
IIIT Bangalore
Bangalore, India
ishmeet.singh@iiitb.org

***Abstract*—Quora is an online platform for questions answering. There are numerous questions everyday. Some of them are asked genuinely and some of them are just put out there to make a statement. Filtering such questions is necessary as it may cause unrest among the Quora audience and may lead to the downfall of the site.**
Separating such sincere and insincere questions manually seems to be a cumbersome task. We are trying to differentiate between such questions and have used traditional machine learning techniques to build a model to tackle this problem.
Our main focus has been to successfully classify the insincere questions. We have tried a lot of different methodologies and have built a model giving fair results. We have tried to understand the open questions of the trends in such insincere questions which have helped us classify them.

***Index Terms*—sklearn, Logistic-Regression, LinearSVM, XG-Boost, LightGBM, RandomForest, SMOTE, train_test_split, F1-Score, SGDClassiier, Bernoulli-NB, Complement-NB, Voting-Classifier, nltk, Stemming, Lemmatization, word2vec, Ada-Boost, Stacking Classifier, Feature Engineering, Grid Search**

I. Introduction

In this world of technological advancements, there is a rise of online social content wherein people share their views and thoughts. There is a lot of online discussion and debates going on about these topics. Such content may get inappropriate when some party forcefully posts something to demean others or to make a statement. There is an urgent need to remove such content. Top social media platforms like Facebook, Quora, Google have these content removed manually. Facebook does this using negative feedback given by the users and then they zero up on the post. Finally, the post is checked upon manually. This is very cumbersome and requires human labour.

In this paper, we have tried to address the Quora Insincere Question Classification problem. This was an earlier competition on Kaggle. We have used this data-set and built some traditional machine learning models to tackle this problem.

The challenge of this problem is to effectively classify the insincere questions in a way that the sincere and genuine questions do not get affected.

We have provided a quantitative analysis of a large corpus of online questions posted on Quora. This corpus was posted by Quora which were some original questions posted on this platform. We have analyzed the findings of a lot of different papers and build up our models based on these.

An insincere question is defined as a question intended to make a statement rather than look for helpful answers. Some characteristics that can signify that a question is insincere are:

- Has a non-neutral tone.
- Has an exaggerated tone to underscore a point about a group of people.
- Is rhetorical and meant to imply a statement about a group of people
- Is disparaging or inflammatory
- Suggests a discriminatory idea against a protected class of people, or seeks confirmation of a stereotype
- Makes disparaging attacks/insults against a specific person or group of people
- Based on an outlandish premise about a group of people
- Disparages against a characteristic that is not fixable and not measurable
- Isn't grounded in reality
- Based on false information, or contains absurd assumptions
- Uses sexual content (incest, bestiality, pedophilia) for shock value, and not to seek genuine answers

II. Related Work

The problem of this question classification is that there is no clear distinction between the patterns of a sincere and insincere question. It is somewhat hard to do it on the human level. Also such distinction between questions requires unbiased opinions which is very hard to find in a human. This problem that we are currently working on has been a competition that was already hosted on Kaggle.

In the original contest, deep learning techniques had been used to tackle this problem. Neural networks such as LSTMs and RNNs have been used to effectively classify and bring out the best F1 score. Some language models such as BERT which is developed by Google has been used to effectively classify the questions which have has provided good results.

As this was a course project, we were limited to using traditional machine learning techniques. Some works have been done such as Deepti et al [1] where char n grams have been used based on word2vec. This had shown astonishingly great results for them. However, we could not do these as we could not find any concrete implementation for this paper.

We had also looked upon the works of Mohammad et al [2] where they have used Tf-idf and Bag of Words Model to build classifiers.

III. DATASET

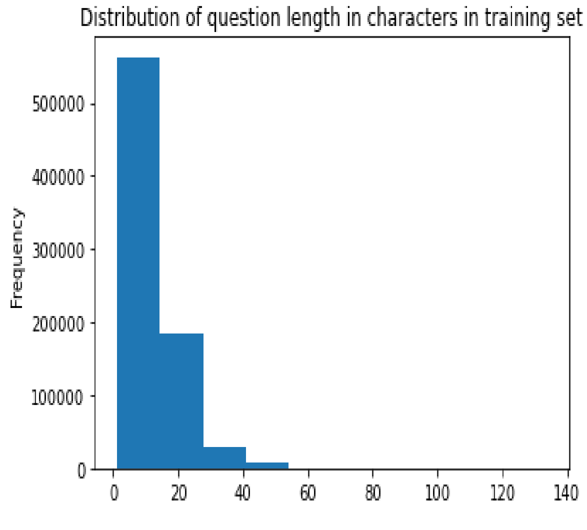
The data set used in this project was provided by Quora in the original contest hosted on Kaggle. The data set has two columns namely 'question_text' which contains the questions and the target value which is binary. It takes values 0 or 1.

There are a total of 783673 rows in the data set to train on. It's a binary classification problem. The scoring metric on which the submissions are judged are on F1 score. The test data has 522448 data points for which we have to predict the binary labels.

IV. OBSERVATIONS

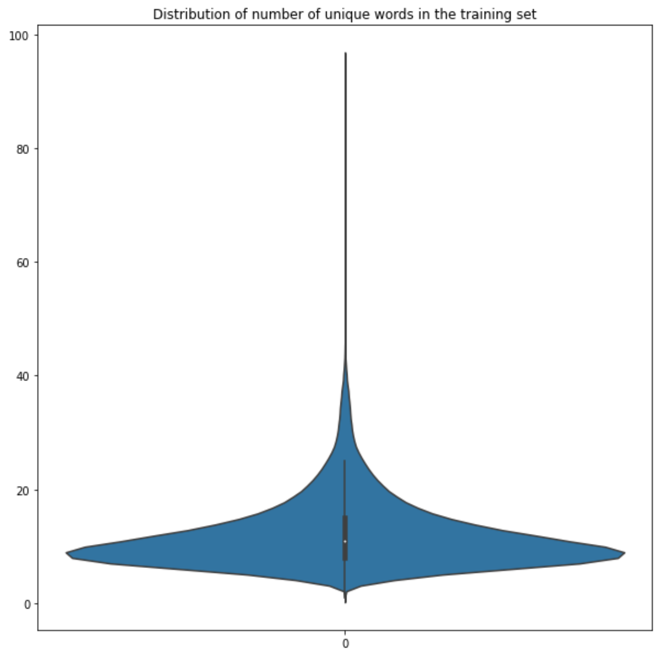
We have visualized our data, before working on it. Visualizing it has provided us with very useful insights.

1) Distribution of data according to the length of the each question.



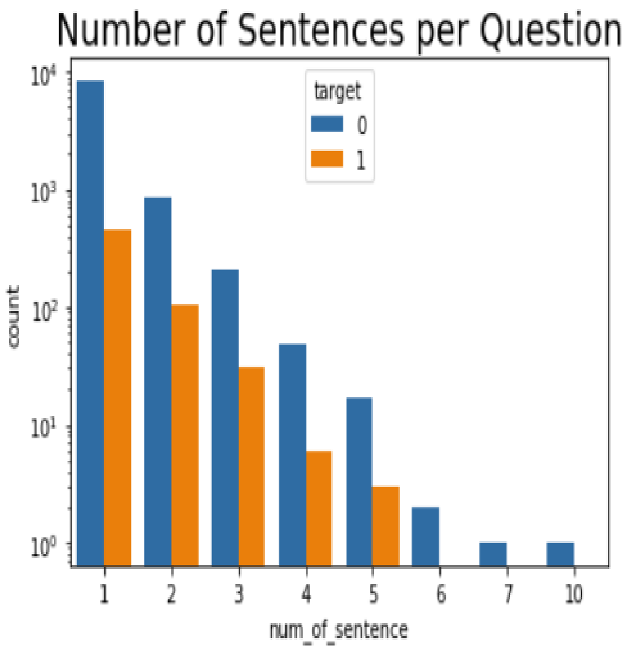
This shows that 99% of the questions have length less equal to 20.

2) Distribution of data according to the number of unique words in the question.



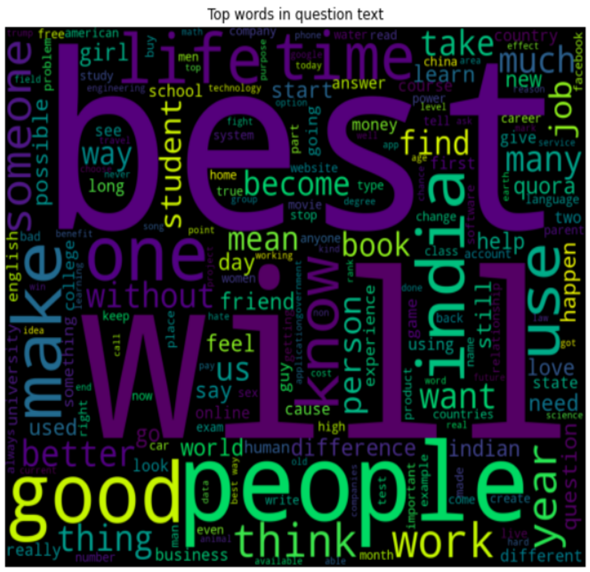
As we can see there are a maximum of 10 - 15 unique words in each question.

3) Count Plot for number of sentences in each question. This has been plotted on the log scale.

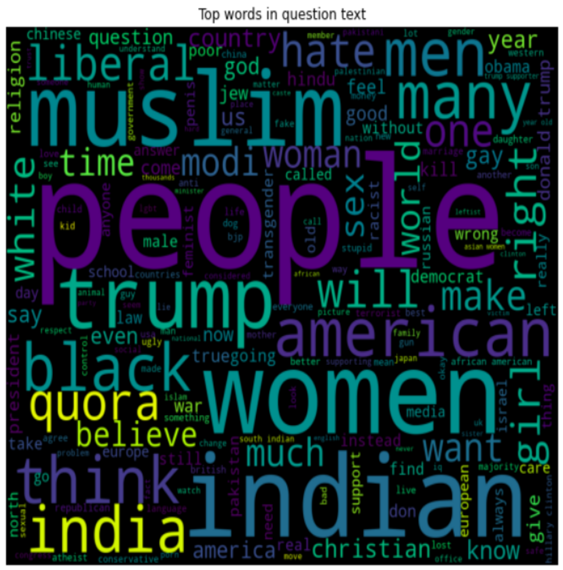


The maximum number of sentence per question was 1.

4) These words are the most common words in Sincere text. The size of the words show their occurrences in the text. Larger sized words are more common in the questions whereas smaller sized words are uncommon.



5) These words are the most common words in Insincere text. As indicated earlier, size is proportional to the occurrences.



V. DATA CLEANING AND PREPROCESSING

To use the data for training we need to first fill in NULL values with appropriate values and change categorical variables to numerical data. This is essential as training a model requires numerical values.

Preprocessing is an essential step in any machine learning problem. The score of a model depends upon it's preprocessing. Good models can perform very poorly if the preprocessing is not good.

We have done lots of preprocessing. For this purpose, we have used packages such as nltk, re. During the initial stages of the project we had done Stemming and Lemmatization. When we tested these with our models, they gave worser scores than if we had not included them.

Stemming - It is the process of reducing inflected (or sometimes derived) words to their word stem, base or root form—generally a written word form

Lemmatization - Lemmatisation (or lemmatization) in linguistics is the process of grouping together the inflected forms of a word so they can be analysed as a single item, identified by the word's lemma, or dictionary form

For text cleaning,

- Firstly, we had replaced some contracted words with their original full forms. For example, shouldn't has been replaced with should not. This is a very essential step as there might be different types of usages for 'should not'. But finally when this is given to the vectorizer and the model they should not treat them as different words and hence this was important.
- Secondly, we had replaced typical misspelling and different spelling words in the question text with correct or common spelling. This is also an important part and gives a good boost in our score. For example, color and colour are two words with US and UK spellings.
- Thirdly, we had replaced some known punctuation with their word form and also added spaces between these punctuation so they can not be grouped with some other word by the vectorizer. Punctuation is also important because some insincere questions have a lot of punctuation such as ** or !!!.
- Finally, we have replaced numbers with hashes.

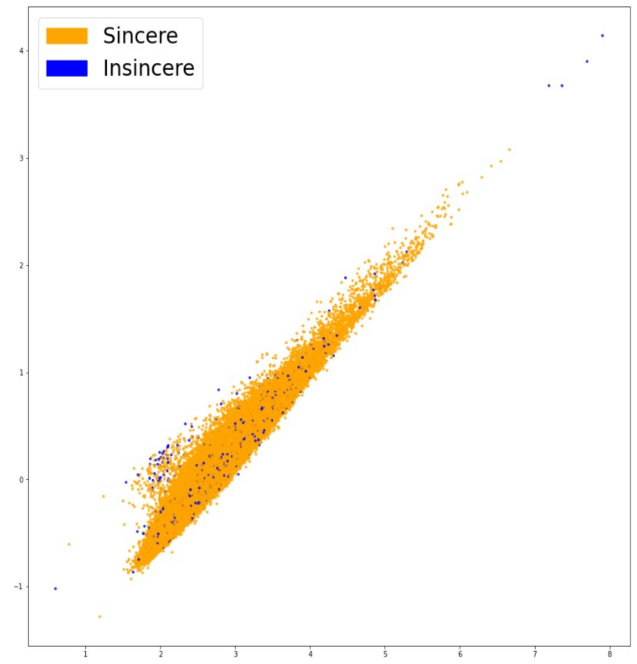
For text preprocessing, we had used tf-idf vectorizer as it worked best with all the traditional machine learning models. *About TF-IDF Vectorizer: It is a numerical statistic that is intended to reflect the importance of a word in a document of a collection or corpus. TF-IDF vectors are calculated in the following way:*

1) *Calculate term frequency (TF) in each document*
2) *Calculate the inverse document frequency (IDF): Take the total number of documents divided by the number of documents containing the word.*
3) *Iterate each document and count how often each word appears*
4) *Calculate TF-IDF: multiply TF and IDF together*

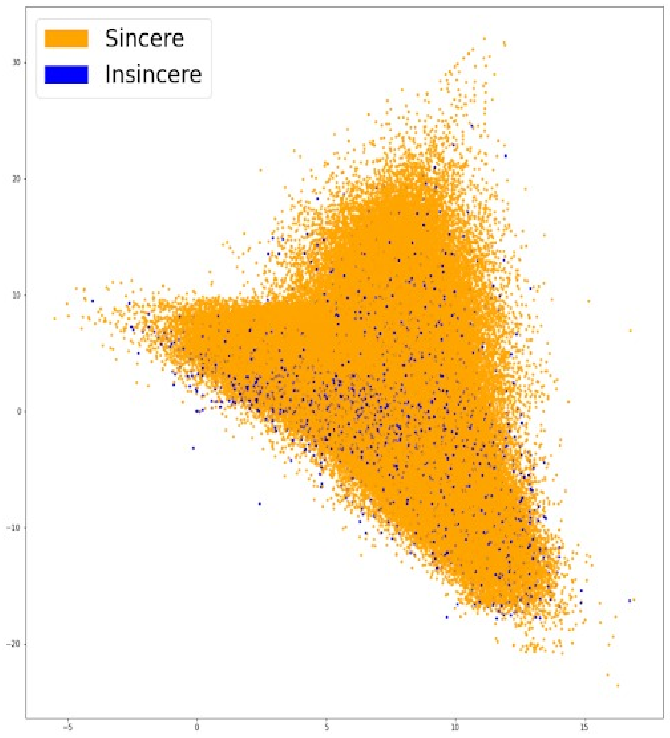
For this tf-idf vectorizer we used two types of analyzers, word and character level. For word level we had set n-grams to (1,3) and for character level we had set the n grams to (1,4). This gave us a good boost in our score because n grams helps to get more context in the data.

We had also used other algorithms such as Word2Vec, Fasttext, Doc2Vec. Word2Vec did not work well with the machine learning models as the traditional models fail to capture context between different vectors. Neural Networks work best with Word2Vec. We could not use Doc2Vec as our notebook crashed everytime. Problem with fast-text pretrained embedding were the Facebook Vectors were 9.7GB in size and every time our notebook crashed when we used that in our project. So we had trained fast-text embedding using our training data which gave us fair scores on the test data.

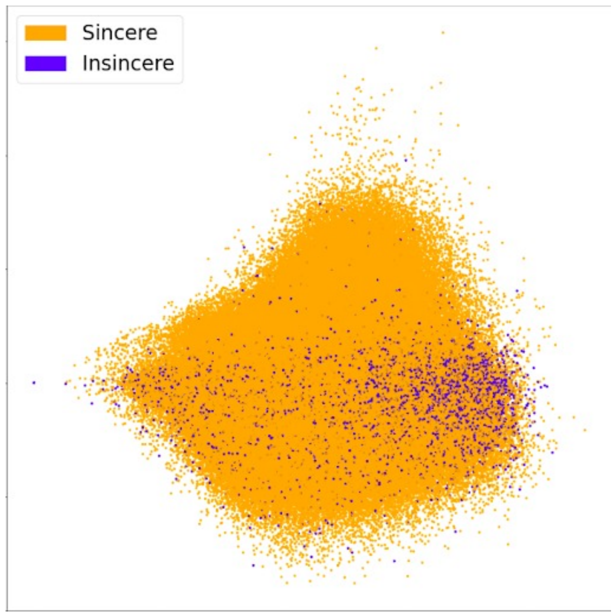
Word2Vec visualization in 2D using PCA



Fasttext visualization in 2D using PCA



Tf-idf visualization in 2D using PCA



So finally we stuck with tf-idf word and char level analyzer which gave us our leader board scores.

VI. Model Selection

We had worked with a lot of different models and oversampling methods as there was bias in our data as mentioned in the Observations section.

With individual classifier we had started training with Logistic-Regression which gave us a pretty good baseline score to work upon. Then we tried to get the best of our preprocessings using this Logistic-Regression Model. As mentioned earlier, Fasttext gave us a fair score in all the work embedding approaches after Tf-idf. We had also done some feature engineering to extract extra columns such as length, number of exclamation marks, etc.

Mainly we had used Logistic-Regression, SGDClassifier, Naive Bayes, RandomForest Classifier, XGBoost, LightGBM models for all our different kinds of data.

Why we could not use newly engineered features with tf-idf? We had originally though of concatenating the newly engineered features with the tf-idf generated vectors. But the problem lies in the sparsity and dimensions of tf-idf vectors. The tf-idf based data-frame in total has 58000 columns and has 783673 rows. This is stored in a memory efficient way in the RAM by pandas. Concatenating this with a dense matrix leads to an implicit conversion of the tf-idf vectors to a dense matrix which leads to crashing of the notebook on Kaggle. So we had to think of some other way of doing this and we thought of combining this with our Tf-idf based model using weighted average. But there was a catch here too as the the threshold for maximum F1 score in both the preprocessing were different and hence it was not being of any use. So we dropped this idea. However, we have documented the performance of the models in the following table for the feature engineered based models. The tree based boosting models such as XGBoost and LightGBM models worked well on this data.

TABLE I: Results of Various Models

Model	Accuracy on Train Set	F1 Score on Test Set
Logistic-Regression	0.938004	0.254101
SGDClassifier	0.938116	0.249538
RandomForest	0.938423	0.268596
BernoulliNB	0.938007	0.179551
LightGBM	0.941106	0.282127
XG-Boost	0.962856	0.285109
Voting-Classifier of XGBoost and LightGBM	0.973754	0.285956

Why we could not combine Fasttext based models and Tf-idf?
As mentioned earlier, fast-text based model had different threshold for the F1 score and hence could not boost our original Tf-idf model. We had primarily used Logistic-Regression, SGDClassifier for Fasttext. We also used the Stacking-Classifier which gave us a good boost to our F1 score. The performance of different models are documented below.

TABLE II: Results of Various Models

Model	Accuracy on Train Set	F1 Score on Test Set
Logistic-Regression	0.94634	0.5143
SGDClassifier	0.945503	0.5089
Stacking-Classifier of Logistic-Regression and SGDClassifier	0.9468	0.5208

Approaches to tf-idf based models
For the tf-idf based model we had trained models such as Logistic-Regression, SGDClassifier, BernoulliNB and RandomForestClassifier. The linear models such as Logistic-Regression and SGD outperformed the tree based and NB models.

We had tried using SVM based models with Linear and RBF Kernel but the training time of these models are so huge that they exceeded the 9 hour notebook limit on Kaggle. Then we had resorted to Nystroem model for Kernel Approximation. It basically reduces the dimensions of the data and had tried it with both LinearSVM and SGD Classifier with hinge loss. But both models did not perform nearly as well as the other models.

The BernoulliNB model was giving us totally surprising results. This model can only work on binary data where the features have value 0 or 1. We have to set the threshold for approximation to 0 or 1 inside the model. Setting this to a value of 0.2 gave us surprisingly good results wherein it performed close to the RandomForestClassifier. This helped us to get a good boost in our leaderboard score after we used it in ensembling.

The only tree based model which gave us a good score was the RandomForestClassifier. The other models such as XGBoost, LightGBM could not be trained as our notebook crashed due to some unknown reasons when we used these. We also trained Ada-Boost model but it did not give us very good scores, so we did not use it in our ensembling.

TABLE III: Results of Various Models

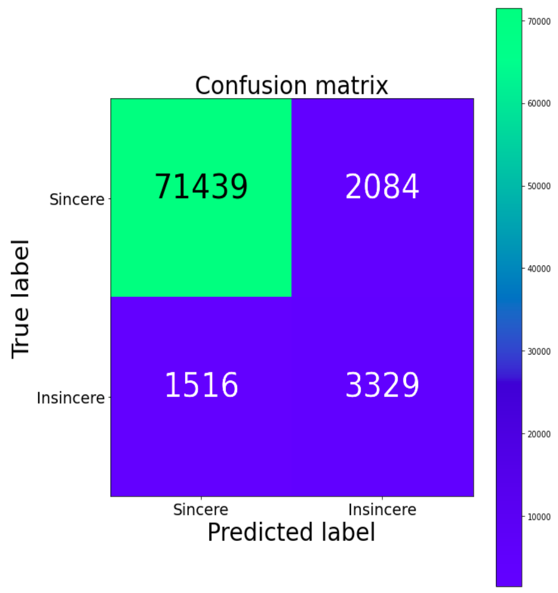
Model	Accuracy on Train Set	F1 Score on Test Set
Logistic-Regression	0.96109	0.6437
SGDClassifier	0.95413	0.6291
RandomForest	0.98666	0.56476
BernoulliNB	0.92799	0.5429
V1	0.9611	0.64811
V2	0.9634	0.6492

V1 here refers to soft voting classifier of BernoulliNB, SGD, Logistics Regression
V2 here refers to soft voting classifier of BernoulliNB, SGD, Logistics Regression, RandomForestClassifier

Here the accuracy is on the train set and the other metrics are calculated on the 90% train set and 10% test set which we have splitted beforehand

All of the above models were trained on Kaggle Notebooks given the limitation of computation power on our own laptops.

CONFUSION MATRIX ON TEST SET



VII. Training Method

In all the models we have first split the training data into 2 parts. 90% of the training data is used for training the model and 10% is used for testing it. This is done by using sklearn's train_test_split. The classes have been equally divided into the train and test set for maintaining consistency.

Earlier we were using Kfold for cross-validation but as the complexity and training time of the models increased, Kfold took a lot of time so we switched to train_test_split. The 10% of the training data was an important criteria to judge our model based on previous models' performances. Also, the notebook was crashing when we used some of the models.

VIII. Conclusion

We were able to build an efficient model for insincere questions classification. This project has huge application in real life and it is very important to tackle such issues.

IX. Challenges and Future Scope

The main difficulties in solving such a problem is to not misclassify the sincere questions. The toxicity of online questions might depend on nationality, ethnic background of the users, cultural background etc. Also these questions are labelled with the help of reviews given by the online users. So sometimes it might be difficult to label them as most of times it depends on the users how they visualize these questions. Though Machine learning is not always the best approach but it always gives the generalised results with fated complexity.

The data also contains noise, questions that are not classified correctly by humans, some sort of screening could be applied to get rid of the noise. We are supposed to use traditional machine techniques otherwise we can also apply RNN (Recurrent Neural Network), BERT, GRU etc. Also we could add more features like location, nationality, religion of the commenters and victim, so that they may help in classifying the questions more appropriately.

Acknowledgment

We would like the thank Professor G. Srinivas Raghavan and Professor Neelam Sinha and all the machine learning TA's especially Tejas Kotha and Nikhil Sai for guiding us along this project and hosting this competition. We have learnt different traditional machine learning techniques that are used to solve problems in real life.

X. Project Link

The code and report can be viewed here

References

[1] D. Mahajan, R Patil, V Shankar, "Word2Vec using Character n-grams"
[2] M Al Ramahi, I Alsmadi "Using Data Analytics to Filter Insincere Posts from Online Social Networks. A case study: Quora Insincere Questions"
[3] Hyper-parameter Tuning for XGBoost and LightGBM Models
[4] S Wang, Christopher D Manning, "Baselines and Bigrams: Simple, Good Sentiment and Topic Classification"
[5] Facebook content's review policy