Quora Insincere Questions Classification

TeamName: deVigNar

Algote Devendhar
MT2020108
IIIT Bangalore
algote.devendhar@iiitb.ac.in

K. Vignaneswar krishna
MT2020107
IIIT Bangalore
kamtam.vignaneswarkrishna@iiitb.ac.in

Duduka Naresh Kumar
MT2020149
IIIT Bangalore
naresh.duduka@iiitb.ac.in

***Abstract*—Community Question Answering has seen a spectacular increase in popularity in the recent past. It has answers to most of the questions, and even if answer is not found, there are sites where you can ask any question and people from all over the world answer it. Quora is one such question answer website. Here anyone can ask question from any domain, which are then answered by others. People misuse this boon and create a havoc by asking questions that are not to be asked in a public forum. The flagging of insincere questions on internet forums is an ongoing challenge for online communities that rely heavily on the faith and support of their users.**

This document is a detailed report on our work on Quora insincere questions classification. We are presenting a model using traditional machine learning methods to tackle insincere and spam contents on the quora. We trained and compared various machine learning models, evaluated different preprocessing and analysis models.

***Index Terms*—Feature Engineering, TFIDF Vectorizer, Word Embeddings, Cross Validation, Logistic Regression, LighGBM, Naive Bayes, SVM, Random Forest Classifier, Voting, Ensamble Techniques.**

I. Introduction

Nowadays, we can not think of the absence of social media in our lives. With so many social media platforms existing on the internet today, the amount of content created online on a daily basis is growing rapidly as more of our world becomes digitized [4]. With the advent and popularity of sites like Yahoo! Answers , Cross Validated, Stack Overflow , Quora , more and more people now use these web forums to get answers to their questions. These forums give people the ability to post their queries online, and have multiple experts and anyone across the world answer them, while being able to provide their opinions or expertise to help other users, a quality that encourages more participation and consequently has led to their popularity. They can also post information that is entirely fake or insincere.

Identifying and eliminating 'toxic' content has become a problem for any major website today. Quora is a website that encourages people to ask questions and learn from each other [6]. However, they also face a challenge of removing insincere questions - questions that are founded on false premises, or that intend to make a statement rather than look for helpful answers. They want to tackle this problem so that their users can feel safe sharing their knowledge.

In this work , an attempt has been made to filter out malicious content by proposing classification models to classify insincere questions. We examine the dataset provided by Kaggle website [1] regarding Quora Insincere Question Classification, Supervised Learning Models are implemented for classification of the questions. The models developed include Naive Bayes, Logistic Regression, Support Vector Machine (SVM), an ensemble of Naive Bayes and Logistic Regression and LGBM and used the different types of word embeddings. Some of the models provide good F1 score.

The report is organized as follows: Section II, we talk about the Problem statement and DATASET. Section III focuses on the EDA (Exploratory Data Analysis), better insights of the data through visualizatoins using various plots. Section IV describes the Preprocessing and Feature Engineering, how the data was prepared and represented, what word embeddings were used, how the imbalance in the dataset was handled. In section V, We also explain the models used and it's selection, how these models work, how we implemented the model with different experiments, and what all evaluation criteria was used to evaluate the model's performance. In section VI, we present the results obtained using various word embedding and classifiers. In section VII, we discuss these results and provides a summary of the work done.

II. Dataset

The dataset that has been used for our project is the Quora Insincere Question Classification Dataset that has been used in the Quora Insincere Question Classification Competition posted on Kaggle this year [1].

The dataset consists of questions in English that users have posted online on Quora. The target values are 0 or 1, corresponding to weather the question should be classified as 'sincere' or 'insincere respectively

The size of the dataset is massive, with the training dataset named 'train.csv' containing 783672 rows. Each data point in the training data consists of 3 columns, namely:

- qid : Unique question identifier
- question_text : Quora question text

- target : a question labeled ”insincere” has a value of 1, otherwise 0

The testing dataset named ‘test.csv’ containing 522449 rows with 2 columns namely qid and question_text for which the labels are not released.

Some examples of insincere questions include:

- Why do girls become escorts or sex workers?
- Why do Chinese hate Donald Trump?

Some examples of sincere questions include:

- What is your free will?
- What is the expected cutoff of JEE Mains 2018?

Our experimentation requires the use of a training dataset, where the information from it is used to train the model. The trained model is then used to predict the value of ‘target’ in the test dataset.

III. Exploratory Data Analysis

Before getting into preprocessing and feature engineering, it is very important to get to know the distribution of data and EDA in order to get better insights of the data. We are presenting a few of those here:

A. Data Distribution

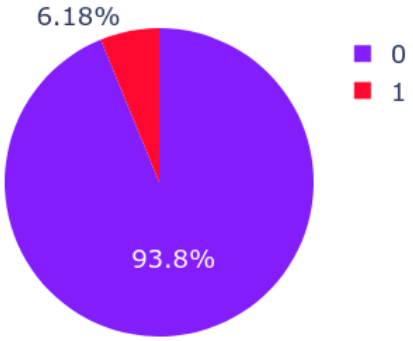

Fig. 1. Class label distribution.

We observed that there is a high imbalance in the two classes. We only have approximately 6% of data that is classified as insincere & 93.8% as sincere.

B. Word Analysis

Word Cloud for target classes. We need to go deep dive into what kind of data we are working with.

We can clearly observed that the most frequently occurred words in the insincere questions were offensive words like – trump, women, muslims, indian etc. and in the sincere questions were – best, will, people, good etc.

C. Other observations

We can see below visualizations through boxplots.There is a slight difference between the distribution of the number of characters, number of punctuations and number of uppercase characters for sincere and insincere questions.

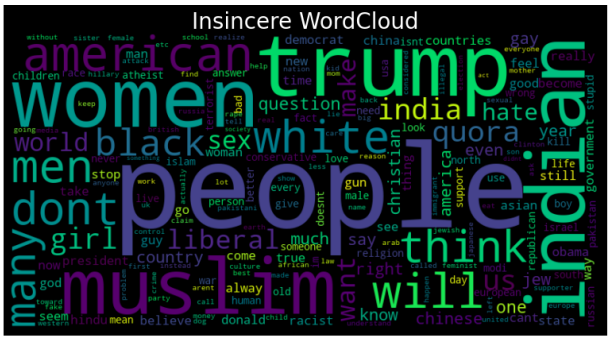

Fig. 2. Wordcloud for Insincere questions

Fig. 3. Wordcloud for Sincere questions

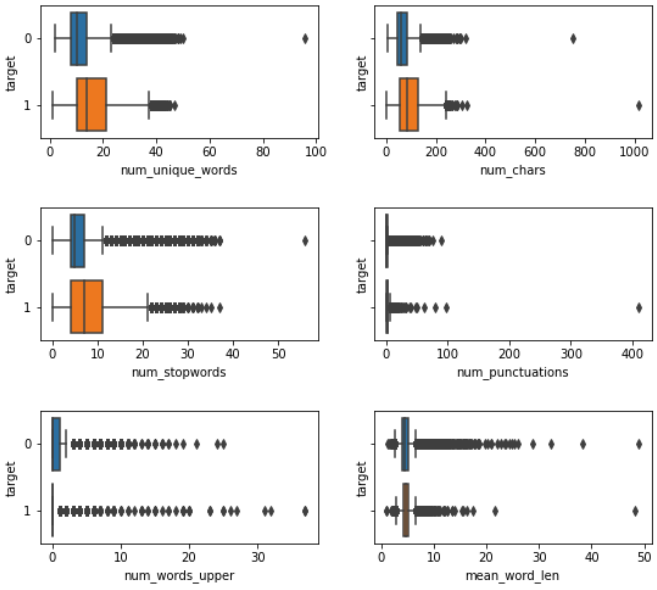

Fig. 4. Data Analysis on questions

IV. Preprocessing and Feature Engineering

A. Data Preparation

Preprocessing plays a vital role in tasks where the input data is in textual format. Since we are working with text data, different preprocessing steps need to be taken before we can feed it to the algorithm. These steps include:

1) Tokenization: The first step for preparing data was breaking up text into individual words called tokens. It is the most essential step for text processing. The sentences are divided into words. The Natural Language Toolkit (NLTK) package is used for this purpose.

e.g., ['Word embeddings plays crucial role.'] becomes ["Word" , "embeddings", "plays", "crucial", "role", "."]

2) Replacing Contractions: Using a pre-defined contractions dictionary map to expend contractions, e.g. replacing "shouldn't" with "should not", "can't" with "can not" .

3) Removing Punctuations, Special characters: We need to carefully choose the list of punctuation which we are going to discard based on need. You can also add other special symbols to list which you want to discard like e.g., " , ? . It's better to handle special characters in text preprocessing step that will helps the model to improve the performance.

4) Converting to Lowercase: Converting all alphabet characters to lowercase e.g. replacing "Indian" with "indian".

5) Stopwords removal: Stop words are common words that do not contribute much of the information in a text document. Words like 'the', 'is', 'a' have less value and add noise to the text data. There is an in-built stopword list in NLTK which we can use to remove stop words from text documents. However this is not the standard stopwords list for every problem, we can also define our own set of stop words based on the domain.

6) Lemmatization: It is a process of replacing the word with its root word using a known word dictionary e.g, "eating" as "eat".

B. Feature Engineering

Besides the data preprocessing steps described in Section 4(A), extra count features (meta features) are added. These features are common in Text Mining classification tasks and count the number of: words, capital letters, and punctuation in the text. Those features are :

- Length of the question
- Number of Words
- Number of Unique words
- Number of Stopwords in the question
- Number of Upper case words
- Number of Punctuations in question
- Number of Special characters

Apart from those meta features, we added other feature as Presence of Sincere/Insincere word in the question. We took the top 10 words from the Insincere wordcloud and Sincere wordcloud and make those words as columns (trump, people, women, muslim, good, think etc) . If question_text contains that word, we make it as 1; otherwise 0.

After feature extraction, the extracted features are stacked with the word embeddings of the question_text, before we feed them to the model using the scipy's hstack [8]. From our observation we found that these meta features did not show significant improvement in our results.

C. Word Embeddings

It is obvious that every mathematical system or algorithm needs some sort of numeric input to work with. Word Embeddings are a method of extracting features out of text so that we can input those features into a machine learning model to work with text data. Formally, a word embedding is a function Words $\rightarrow \mathbb{R}^d$ that maps words to real valued vectors of a fixed dimension (Bengio et al., 2003). Many authors have reported that word embeddings perform surprisingly well for text classification tasks (Reimers and Gurevych, 2019). We used the some of the pretrained word embeddings also like Glove. The word embedding that we have worked with for this dataset are:

*1) **Count Vectorizer**:* It Considers a Corpus C of D documents d1,d2.....dD and N unique tokens extracted out of the corpus C. The N tokens will form our dictionary and the size of the Count Vector matrix M will be given by D X N. Each row in the matrix M contains the frequency of tokens in document D(i).

*2) **TF-IDF vectorization**:* 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:

i) Calculate term frequency (TF) in each document.

ii) Calculate the inverse document frequency (IDF): Take the total number of documents divided by the number of documents containing the word.

iii) Iterate each document and count how often each word appears.

iv) Calculate TF-IDF: multiply TF and IDF together [7].

To extract the features from the comments we used TF-IDF vectorizer over the combined text of train and test in order to capture all the words occurring in the data for an n-gram of range (1, 4).

*3) **Glove**:* GloVe was developed by Pennington, et al. at Stanford [2]. It is called Global Vectors as the global corpus statistics are captured directly by the model [3]. We used GloVe embedding's which is based on factorizing a matrix of word co-occurrence statistics. We experimented on all dimensions of glove vectors and decided to work with 300-dimensional GloVe vectors. To compute a GLOVE embedding

for an question_text we averaged the single GLOVE embeddings of all words contained in the question_text.

*4) **Word2Vec**:* Word2Vec is an algorithm invented at Google for training word embeddings. It relies on the distributional hypothesis. The distributional hypothesis states that words which, often have the same neighboring words tend to be semantically similar. Word2Vec models are shallow neural network with an input layer, a projection layer and an output layer. It is trained to reconstruct linguistic contexts of words.

It comes in two flavors, the Continuous Bag-of-Words model (CBOW) and the Skip-Gram model. Algorithmically, these models are similar, except that CBOW predicts target words (e.g. 'mat') from source context words ('the cat sits on the'), while the skip-gram does the inverse and predicts source context-words from the target words.

*5) **TF-IDF character n-gram approach**:* We follow a bag-of-words (BoW) model for extracting TFIDF features from the character n-grams contained within question_text [5]. We use an n-gram range between 2 and 5-grams. Bag of n-grams enables using partial information about structure of the text. TF-IDF character n-gram with range (2, 5) based model performs better for us than word embedding-based models.

D. Resampling Techniques

We can clearly observed that from the Section 3(A), we have an imbalance in the dataset. The learning phase and the prediction of machine learning and deep learning algorithms can be affected by the problem of an imbalanced dataset. The decision function of the models might favor the class with the larger number of samples.

*1) **Undersampling**:* We need to down-sample the majority class, i.e., the sincere class, to equal the size of the minority class, i.e., the insincere class. This will help us balance the two classes in the dataset as well as decrease the processing time as the number of samples in the training data will also decrease. We used the under-sampling techniques provided by imbalanced-learn [12].

After Undersampling techniques, we have seen huge reduce in the size of the training dataset nearly becomes the 90k datapoints. So It did not work much for us in improving the results.

*2) **Random Oversampling**:* One approach to addressing imbalanced datasets is to oversample the minority class. The simplest approach involves duplicating examples in the minority class, although these examples don't add any new information to the model. Instead, new examples can be synthesized from the existing examples.

This is a type of data augmentation for the minority class and is referred to as the *Synthetic Minority Oversampling Technique* or *SMOTE* for short [11].

By above oversampling technique, we have observed the very huge increment of minority class datapoints. And dataset becomes the near 1400k. With this huge dataset, indirectly we are insisting the model become overfitting rather than generalized model. From our results, what we observed is there is a significant decrease in the F1 score of the model, with which we trained using the oversampled data.

V. Model Selection

Once done with pre-preprocessing the data and feature extraction, We experimented a number of different machine learning models for the purpose of classification. We used F1 Score as the sole evaluation metric in the project and applied this metric to all the used models. As for the models in particular, we made use different models that will discuss below.

A. Naive Bayes

This model is based on Bayes theorem with the naïve assumption of independence between each pair of features. If we need to classify the vector X = x_1 .... x_n into m classes, C_1 ...C_m . We need to find the probability of each class given X. Then we can assign X the label of the class with highest probability. The probability is calculated using Bayes theorem which is defined as:

$$P(C_\mathrm{i} \mid X) = \frac{P(X \mid C_\mathrm{i})P(C_\mathrm{i})}{P(X)}$$

We used the Naïve Bayes classifier for multinomial models provided by sklearn [9] with default parameters. We chose the multinomial Naïve Bayes classifier because is appropriate for text classification. For this classifier we have used Count Vectorizer and the TF-IDF vectorizer and the got the 0.564 and 0.551 F1 Scores respectively.

B. Logistic Regression

Logistic regression is most commonly used when the data in question has binary output, so when it belongs to one class or another, or is either a 0 or 1. When using logistic regression, a threshold is usually specified that indicates at what value the example will be put into one class vs. the other class. Although logistic regression is best suited for instances of binary classification, it can be applied to multiclass classification problems. Logistic regression makes use of the formula

$$h_\theta(x) = g(\theta^\mathrm{T} x) = \frac{1}{1 + e^{-\theta^\mathrm{T} x}}$$

where x represents features. We have done different experiments with the logistic regression algorithm such as using different types of embeddings like Count Vectorizer, Tf-idf Vectorizer with Word level and Char level and Glove embeddings with hyperparameters tuning. We have tuned the different hyperparameters also, and among them we got the highest F1 score 0.62467 with the Char level n-grams using Tf-idf Vectorizer.

C. *SVM*

Support vector machine works by finding and constructing a hyperplane in N-dimensional space that separates the points between two classes, N being the number of features. The hyperplane is determined by finding a plane that has the maximum margin which is the distance between the data point of two classes. Points that fall on the side of the hyperplane can be attributed to different classes. We used the Support Vector Classifier provided by sklearn [10] for training and testing. The kernel type to be used in the algorithm is 'linear' (x, x'). We have tried Tf-idf and Glove embeddings with SVM, what we observed is that it converging slowly and taking lot of time. We got the 0.60494 F1 score.

D. *LightGBM*

It is a gradient boosting framework that uses tree based learning algorithms. It is considered to be a fast processing algorithm. Compared to other algorithms where trees grow horizontally, meaning it grows level-wise, LightGBM trees grow vertically. It is designed to be distributed and efficient with the many advantages. The advantages of LightGBM does better than the other algorithms is that in case of massive datasets, like the one used in our implementation. The light in LightGBM comes from the fact that it runs at a very high speed, and consumes a lot less memory. It also has support for GPU learning.

We tried it with the Tf-idf Vectorizer and have set 5 folds for our implementation, we got the 0.59927 F1 Score.

E. *Random Forest Classifier*

Random forests or random decision forests are an ensemble learning method for classification, that operate by constructing a multitude of decision trees at training time and output the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. We used it in our ensemble techniques.

F. *Ensembling Techniques*

Ensemble methods is a machine learning technique that combines several base models in order to produce one optimal predictive model. It is well-known that ensemble methods can be used for improving prediction performance.

After training the model with these different type of algorithms. Then we started looking into various ensembling techniques like VotingClassifier, EnsembleClassifier and BaggingClassifier after reading an article on MLWave.com which discusses about ensembling guides in detail [13]. We tried all of the above mentioned ensemble techniques with the previously mentioned algorithms like Logistic Regression, RandomForestClassifier, MultinomialNB. Given the complexity and time consumption we shifted most of our model experimentation and training to Kaggle and Google Collab Kernels.

We have used voting classifier, A Voting Classifier is a machine learning model that trains on an ensemble of numerous models and predicts an output. It simply aggregates the findings of each classifier passed into Voting Classifier and predicts the output class based on the highest majority of voting. The idea is instead of creating separate dedicated models and finding the accuracy for each them, we create a single model which trains by these models and predicts output based on their combined majority of voting for each output class. Here we have given a higher weight to our best performed model till now, i.e., logistic regression, then to naive bayes, least weight to the output generated by the Random forest classifier. The F1 Score we got is 0.62323 .

G. *Evaluation Metrics*

We chose F1 Score as a evaluation metric. It is the harmonic mean of the two. It results nearly the weighted average of Precision and Recall. Therefore, this score takes both false positives and false negatives into account.

F1-Score :

$$F1Score = \frac{2 * precision * recall}{precision + recall}$$

where,

$$precision = \frac{TP}{TP + FP}$$

$$recall = \frac{TP}{TP + FN}$$

VI. Results and Discussion

We evaluated different classifiers on each one of the feature matrices resulting from each data preprocessing and representation. With the Baseline models we got less accuracy than the further models. Firstly, Naive bayes algo gave us good results when we used Count Vectorizer than the Tf-idf Vectorizer. After that we tried with the Logistic Regression with different embeddings like Count, Tf-idf and Glove embeddings and some basic preprocessing techniques. Char level n-gram embeddings using Tf-idf Vectorizer with Logistic regression gave us the better F1 score overall. For us excessive use of preprocessing techniques lead to the decrease in the F1 score than the normal. Again we tried with SVM, LightGBM models. Later We looked into the ensemble techniques with the previously mentioned algorithms like Log Regression, Naive Bayes and Random Forest Classifier with the voting classifier.

Below are the results for the different machine learning models, with different embedding

TABLE I
RESULTS OF DIFFERENT MODELS

S.No	*Classifier*	*F1 Score*
1	Naive Bayes + Count Vectorizer	0.56418
2	Naive Bayes + Tf-idf Vectorizer	0.55926
3	Logistic Regression + Count Vectorizer	0.57992
4	Logistic Regression + Tf-idf + Metafeatures	0.60346
5	Logistic Regression + Tf-idf Vectorizer	0.61132
6	Logistic Regression + Glove	0.46351
7	Logistic Regression + Glove + Oversampling	0.35113
8	SVM + Tf-idf Vectorizer	0.60346
9	LightGBM + Tf-idf Vectorizer	0.59927
10	Maxvoting(LogReg+NaiveBayes+RandomForest)	0.62323
11	LogReg+ Char Embeddings + Thresholding	0.62467

VII. CONCLUSION AND FUTURE WORK

We were able to analyze the 'Quora insincere Questions Classification' dataset to classify the questions as sincere and insincere. We processed the dataset and implemented the above experiments with the different algorithms.

We observed that above algorithms can achieve that good accuracy for classifying insincere questions without handling the imbalance, i.e., they can learn from an imbalanced dataset. The best results were obtained using Logistic Regression when we are using char level embeddings with Tf-idf along with thresholding and other model is ensemble technique (Logistic Regression + Naive Bayes + Random Forest). However, this cannot be generalized. There could be different results with the different techniques.

In the future, one thing we would be very interested to look into is different character embeddings compared to word embedding along with different Ensemble techniques. We would also like to look into implementing the Neural Networks, LSTM's, and GRU etc.

ACKNOWLEDGMENT

We would like to thank to Professor G. Srinivas Raghavan, Professor Neelam Sinha and our Machine Learning Teaching Assistants, Nikhil Sai Bukka, Shreyas Gupta, Tejas Kotha and Tanmay Jain for giving us the opportunity to work on the project and help us whenever we were struck by giving us ideas and resources to learn from. Their highly detailed lectures on various topics helped us understand what we were actually doing.

We would gladly say that we had a great learning experience while working on the project.

REFERENCES

[1] "Kaggle," [Online]. Available: https://www.kaggle.com/c/quora
[2] "GloVe: Global Vectors for Word Representation". Jeffrey Pennington, Richard Socher, Christopher D. Manning. Computer Science Department, Stanford University. [Online]. Available: https://nlp.stanford.edu/pubs/glove
[3] J. P. a. R. S. a. C. D. Manning, "Glove: Global vectors for word representation," in In EMNLP, 2014.
[4] "Forbes," [Online]. Available: https://www.forbes.com/sites/bernardmarr/2018/05/21/h ow-much-data-do-we-create-every-day-the-mind- blowing-stats-everyone-should-read/3e5bb47c60ba.
[5] TF-IDF Character N-grams versus Word Embedding-based Models for Fine-grained Event Classification: A Preliminary study [Online]. https://www.aclweb.org/anthology/2020.aespen-1.6.pdf
[6] "Quora," [Online]. Available: https://www.quora.com/.
[7] Kaitlyn, "Calculating tf-idf with python," 2017, [Online; posted 3-June-2017]. [Online]. Available:https://methodi.ca/recipes/calculating-tf-idf-python
[8] E. Jones, T. Oliphant, P. Peterson et al., "SciPy: Open source scientific tools for Python," 2001–, [Online; accessed ¡today¿]. [Online]. Available: http://www.scipy.org/
[9] "sklearn Naive Bayes Documentation," [Online]. Available: https://scikit- learn.org/stable/modules/generated/sklearn.naive_bayes .MultinomialNB.html
[10] "sklearn Documentation," [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html
[11] "imbalanced-learn Documentation about oversampling" https://link.springer.com/chapter/10.1007/978-3-030-47436-2_7
[12] "imbalanced-learn Documentation," [Online]. Available: https://imbalanced- learn.readthedocs.io/en/stable/under_sampling.html
[13] A. S. Hendrik Jacob van Veen, Le Nguyen The Dat, "Kaggle ensembling guide," 2015, [Online; posted 6-February-2018]. [Online]. Available: https://mlwave.com/kaggle-ensembling-guide/
[14] Nadbor, "DS lore," Text Classification With Word2Vec - DS lore. [Online]. Available: https://nadbordrozd.github.io/blog/2016/05/20/text-classification-with-word2vec/. [Accessed: 15-Dec-2018].
[15] B. a. L. L. a. V. S. Pang, "Thumbs Up?: Sentiment Classification Using Machine Learning Techniques," in Association for Computational Linguistics, Stroudsburg, PA, USA, 2002.
[16] L. a. Y. M. a. Z. M. a. L. X. a. Z. T. Jiang, "Target- dependent Twitter Sentiment Classification," in Association for Computational Linguistics, Portland, Oregon, 2011.