Quora Insincere Questions Classification

Author: Jeevan Anand Anne
Date: December 14th, 2018

I. Definition

Project Overview

This project aims to identify the insincerity in the Quora questions i.e. to classify whether the Quora questions are insincere or not. Quora is a platform that empowers people to learn from each other. On Quora, people can ask questions and connect with others who contribute unique insights and quality answers. A key challenge is to weed out insincere questions, those founded upon false premises, or that intend to make a statement rather than look for helpful answers. This is very important for very good user experience. [2]

This is a case of classification which is part of Supervised Learning. "target" is label which is a question labeled "insincere" has a value of 1, otherwise 0, while qid and question_text are other variables.

Datasets and Inputs
Dataset Source: https://www.kaggle.com/c/quora-insincere-questions-classification/data [2]

Problem Statement
Predicting whether a question asked on Quora is sincere or not. It is a supervised classification problem.

Metrics
Since this is a supervised classification problem, F1 Score is used for evaluation. Here in this problem both precision and recall are important to consider when we have high class imbalance.

It can be mathematically defined as:

F1 = (2 * P * R)/ (P + R)
where,
P = Precision. It's a proportion of actual positives of all predicted positives.
R = Recall. It's how many of the Actual Positives our model capture through labeling it as Positive (True Positive).

F1 score is a combined metric of precision and recall as mentioned.

II. Analysis

Data Exploration & Visualization

We have train and test datasets. Train dataset has 2 features (qid, question_text) and 1 target variable. Test dataset has 2 features (qid, question_text).

```
#import data
train_df = pd.read_csv("train.csv")
test_df = pd.read_csv("test.csv")
print("Train shape : ", train_df.shape)
print("Test shape : ", test_df.shape)
```

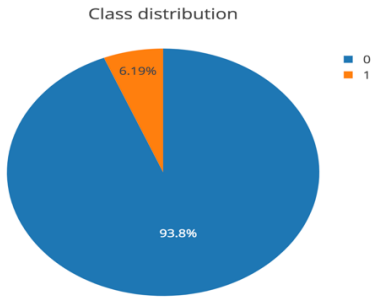
```
Train shape :  (1306122, 3)
Test shape :  (56370, 2)
```

```
#display top 5 rows
train_df.head()
```

	qid	question_text	target
0	00002165364db923c7e6	How did Quebec nationalists see their province...	0
1	000032939017120e6e44	Do you have an adopted dog, how would you enco...	0
2	0000412ca6e4628ce2cf	Why does velocity affect time? Does velocity a...	0
3	000042bf85aa498cd78e	How did Otto von Guericke used the Magdeburg h...	0
4	0000455dfa3e01eae3af	Can I convert montra helicon D to a mountain b...	0

Train dataset has approximately 1.3M observations and test dataset has 56K observations.
We have only unstructured text data from which we will have to extract relevant features to feed into the machine learning algorithms. Preprocessing has to be done on the text.

What is the class (target) distribution?

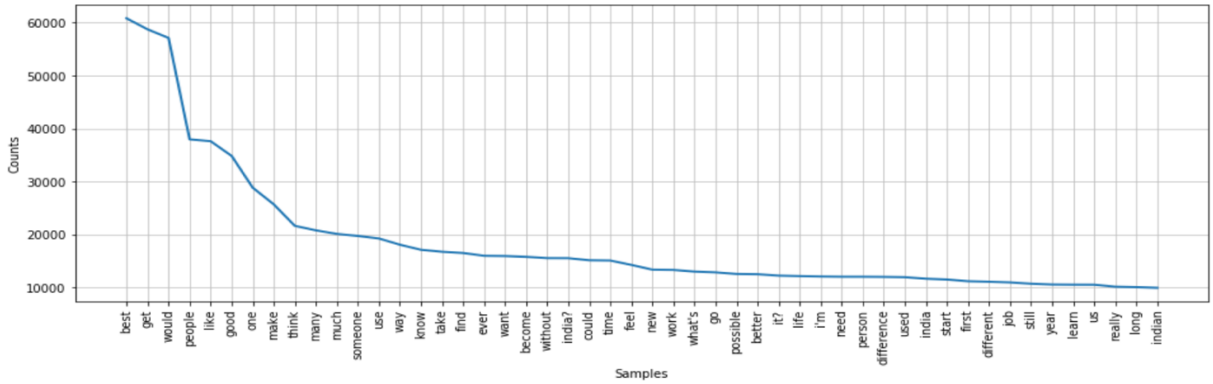


From the plot, we can say that there are almost 93.8% of Quora questions are sincere (target=0) and 6.2% of Quora questions are insincere (target=1), which is heavily imbalanced.

What are the most frequent words in both of the classes?

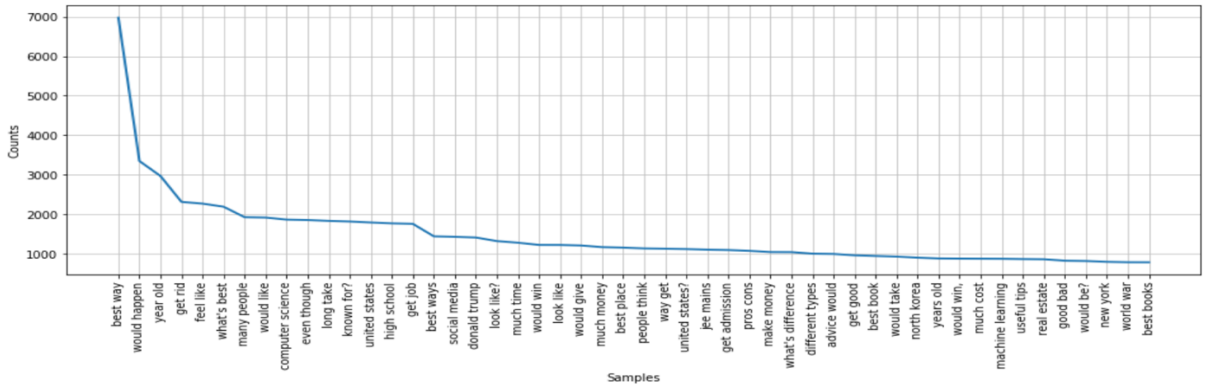
For Sincere Questions:

Unigram:



It makes sense based on the words in the sincere questions we see in the plot.

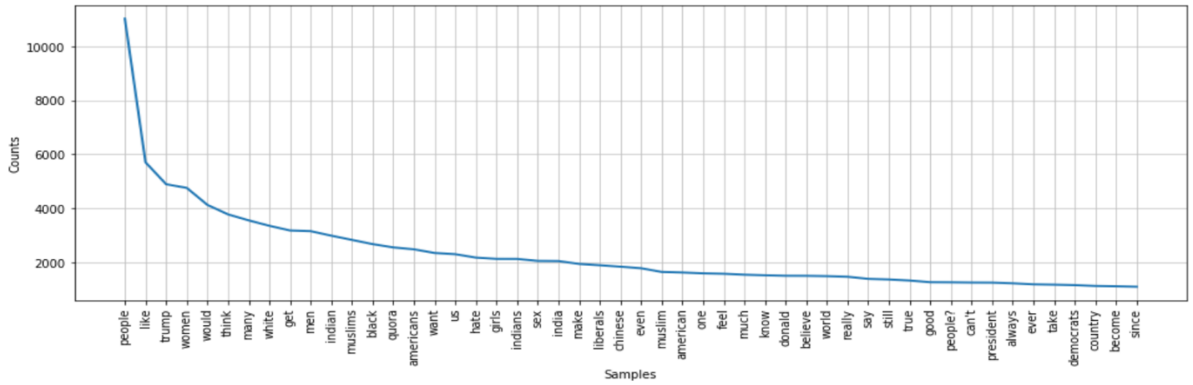
Bigram:



All the words in the sincere questions we see in the plot are all in positive.

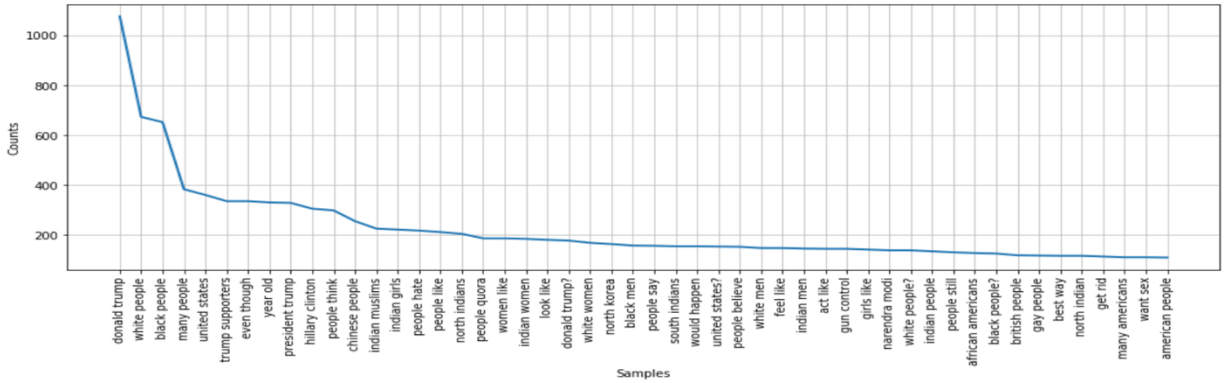
For Insincere Questions:

Unigram:



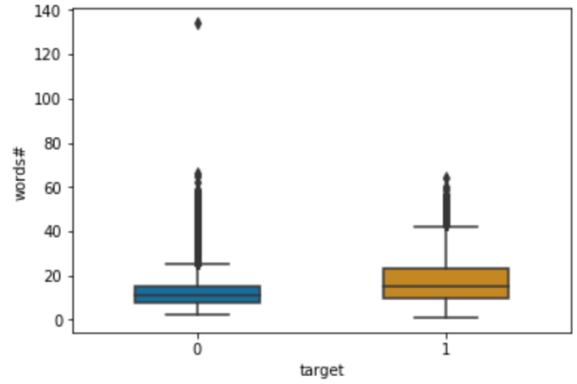
We see most of the frequents words in insincere questions are about politics, race etc.

Bigram:



Now it makes more sense about the words used in insincere questions.

What are number of words in training dataset?



There are a greater number of words for insincere questions than in sincere questions when comparing the median values. This might be a good feature to use.

Algorithms and Techniques

There are many different choices of machine learning models for classification which can be used to train a model. We will implement following different classifiers for this purpose:

- Linear Classifier

a) Implementing a Linear Classifier (Logistic Regression). Logistic regression measures the relationship between the categorical dependent variable and one or more independent variables by estimating probabilities using a logistic/sigmoid function. One can read more about logistic regression here

Benchmark

The benchmark is the F1 score of the logistic regression classification technique used, which obtained a F1 score of ~0.365 using pipeline which includes tfidf vectorization, other numerical features like number of words, number of unique words without any cross validation. We will try to beat this leaderboard score on Kaggle.

Methodology

Data Preprocessing

Stopword Removal

A stop word is a commonly used word (such as “the”, “a”, “an”, “in”) that a search engine has been programmed to ignore, both when indexing entries for searching and when retrieving them as the result of a search query.
We would not want these words taking up space in our database or taking up valuable processing time. For this, we can remove them easily, by storing a list of words that you consider to be stop words. NLTK (Natural Language Toolkit) in python has a list of stopwords stored in 16 different languages. You can find them in the nltk_data directory, where you have downloaded.

N-grams

The idea is to make (statistical) predictions about what is happening in a sentence. Things that happen could be that a particular word shows up next, or that an element belonging to a particular word class shows up next. [5]

Here is our sentence "I am doing capstone project for MLND"
The machine wants to get the meaning of the sentence by separating it into small pieces. How should it do that?

- It can regard words one by one. This is unigram; each word is a gram.
  "I", "am", "doing", "capstone", "project", "for", "MLND"

- It can regard words two at a time. This is bigram (digram); each two adjacent words create a bigram.
  "I am", "am doing", "doing capstone", "capstone project", "project for", "for MLND"
- It can regard words three at a time. This is trigram; each three adjacent words create a trigram.
  "I am doing", "am doing capstone", "doing capstone project", "capstone project for", "project for MLND"

TF-IDF

tf-idf, short for term frequency–inverse document frequency, is a numeric measure that is used to score the importance of a word in a document based on how often it appeared in that document and a given collection of documents. The intuition for this measure is: If a word appears frequently in a document, then it should be important, and we should give that word a high score. But if a word appears in too many other documents, it's probably not a unique identifier, therefore we should assign a lower score to that word. The math formula for this measure: [6]

$$tfidf(t,d,D)=tf(t,d)\times idf(t,D)$$

Where t denotes the terms; d denotes each document; D denotes the collection of documents.

The first part of the formula tf(t,d) is simply to calculate the number of times each word appeared in each document.

Let's first write down the complete math formula for IDF.

$$idf(t, D) = log\frac{|D|}{1 + |\{d \in D : t \in d\}|}$$

- The numerator: D is inferring to our document space. It can also be seen as D = d1, d2, ..., dn where n is the number of documents in your collection.

- The denominator: $|\{d \in D: t \in d\}|$ implies the total number of times in which term t appeared in all of your document d (the $d \in D$ restricts the document to be in your current document space). Note that this implies it doesn't matter if a term appeared 1 time or 100 times in a document, it will still be counted as 1, since it simply did appear in the document. As for the plus 1, it is there to avoid zero division.

Creating training and validation datasets from train data, in order to build a model on train data and evaluate on validation dataset.

Implementation

Logistic Regression
Approach1:

I used sklearns Logistic Regression classifier without any cross validation, with parameters C=5. and solver='sag'. Here solver is used for optimization problems, we have selected sag (Stochastic Average Gradient), this optimizer handle L2 penalty. This model was used to predict the test dataset and uploaded to Kaggle. It got the following scores:

Public Leaderboard score of 0.365 on Kaggle and there will be no private leaderboard until the competition is over.

Approach2:

I used sklearns Logistic Regression classifier with kfold(k=5) cross validation, with the same paramters mentioned above.

This model was used to predict the test dataset and uploaded to Kaggle. It got the following scores:
Validation Score: 0.597
Public Leaderboard Score: 0.459

Refinement

Used kfold cross validation with 5 folds, evaluated the model on validation dataset and selected a best F1 score threshold, which made a huge 10% improvement over a simple Logistic Regression with the same parameters.

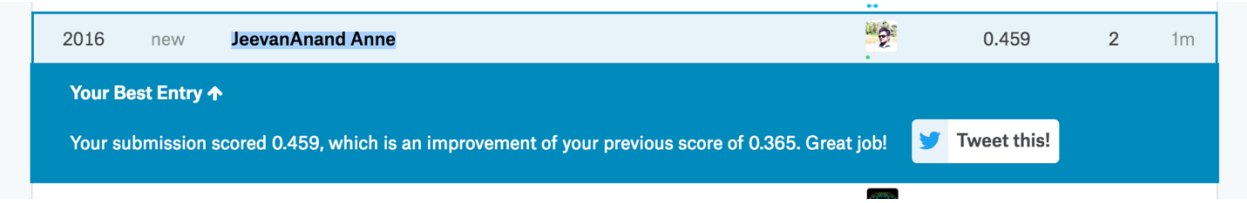
Results

Model Evaluation and Validation

Initially built a simple Logistic Regression model without any cross validation and and used parameters C=5. and solver='sag'. Here solver is used for optimization problems, we have selected sag (Stochastic Average Gradient), this optimizer handle L2 penalty.

Performed 5-fold Cross validation on the whole training dataset using Logistic Regression. F1 score is being used as an evaluation criteria and validation score is 0.597.

```
F1 score at threshold 0.1 is 0.5685067028300838
F1 score at threshold 0.11 is 0.5767330162102249
F1 score at threshold 0.12 is 0.5837238213636798
F1 score at threshold 0.13 is 0.5898637978681406
F1 score at threshold 0.14 is 0.593576234607084
F1 score at threshold 0.15 is 0.5955237896494157
F1 score at threshold 0.16 is 0.5971034630183839
F1 score at threshold 0.17 is 0.5972491865139046
F1 score at threshold 0.18 is 0.59652752537255
F1 score at threshold 0.19 is 0.5945699032175178
F1 score at threshold 0.2 is 0.5933791016988597
```



It almost improved 10% on the public leaderboard on Kaggle.

Justification

Difference in the benchmark and my solution:

Models/Parameters	F1 Score
Final Model	0.459
Benchmark	0.365

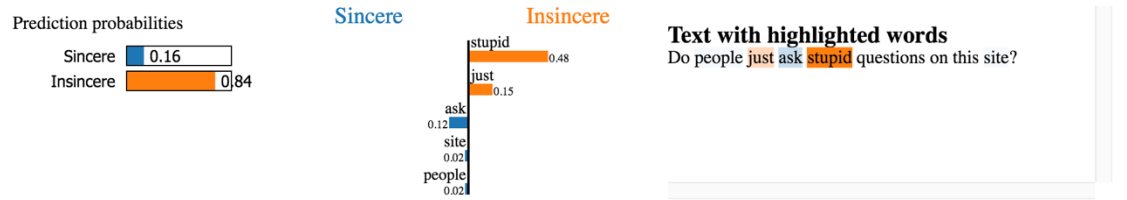
Given that the benchmark I used and created a better model than benchmark, the improvement of about 10% is significant enough to consider my solution a satisfactory one.

Sometimes simple model gives better results.

Conclusion

Free-Form Visualization

```
In [73]: idx=117
exp = explainer_quora.explain_instance(test_df['question_text'][idx], classifier_quora.predict_proba, num_features=5)
exp.show_in_notebook(text=test_df['question_text'][idx])
```




```
In [81]: idx=56368
exp = explainer_quora.explain_instance(test_df['question_text'][idx], classifier_quora.predict_proba, num_features=5)
exp.show_in_notebook(text=test_df['question_text'][idx])
```

With our model built, lets interpret the reasons of the predicted class using LIME (Local Interpretable Model-Agnostic Explanations) [7].

Our model:

- Correctly predicted “Insincere” for the text “Do people just ask stupid questions on this site?” with high probability of the word stupid in the text.
- Correctly predicted “Sincere” for the text “Do you think our system is fair? Consider factors such as the tax code, the welfare state, prejudice/discrimination, and other factors.”

Both the results make sense based on the words appear in the text

Reflection

The whole project experience can be summarized as follows:

- Wanted to explore more NLP preprocessing techniques, and advanced techniques like word2vec and thought this Kaggle competition will help me look at mentioned techniques
- Found the dataset on the Kaggle

- Visualized the data, did preprocessing by learning from other contestants in kernels from Kaggle.
- Preprocessing was an integral part of this project, which helped to much better results.
- Using a seed to reproduce the results was an important discovery.
- Applying selected algorithms and tuning the best performing model.
- Using KFold Cross Validation yields better results.
- Comparing my tuned model against the author's result which I used as the benchmark for this project.
- Interesting part was learning new preprocessing techniques and visualization techniques for creating features.
- Difficult was figuring out to run the bagging/boosting models as it is taking so much of time.

Improvement

1. Create more better features – feature engineering.
2. Preprocessing like contractions, correcting the words which are mistyped etc.
3. Use advanced bagging and boosting models while having better resources GPU for example
4. Implement word2vec with the pretrained embeddings available and improve the coverage of the vocabulary with necessary preprocessing at each stage
5. Implement sequence models like LSTM, GRUs which might help us in improving our score.

References

1) https://www.kaggle.com/c/quora-insincere-questions-classification/kernels
2) Quora Inc, https://www.kaggle.com/c/quora-insincere-questions-classification/data
3) https://www.cnblogs.com
4) https://www.geeksforgeeks.org - NLTK
5) https://en.wikipedia.org/wiki/N-gram
6) https://en.wikipedia.org/wiki/Tf%E2%80%93idf
7) https://homes.cs.washington.edu/~marcotcr/blog/lime/