Quora Insincere Questions Classification

CIS 668/ IST 664 - Natural Language Processing

Professor Lu Xiao

Apurva Sharma
Harsh Doshi
Hemin Vyas
Simaant Patil

Contents:

Introduction and Objective:

Introduction:

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 or may have a rhetorical question rather than look for helpful answers.

Objective:

To classify text and predict whether a question asked on Quora is sincere or not, using NLP

Some characteristics that can signify that a question is insincere:

- 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; isn't grounded in reality
- 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
- Based on false information, or contains absurd assumptions
- Uses sexual content (incest, bestiality, pedophilia) for shock value, and not to seek genuine answers

Representative Technique and Dataset in our Project:

Text classification is a common task in natural language processing, which transforms a sequence of the text of indefinite length into a category of text. One of the widely used natural language processing task in different business problems is “Text Classification”. The goal of text classification is to automatically classify the text documents into one or more defined categories. Some examples of text classification are:

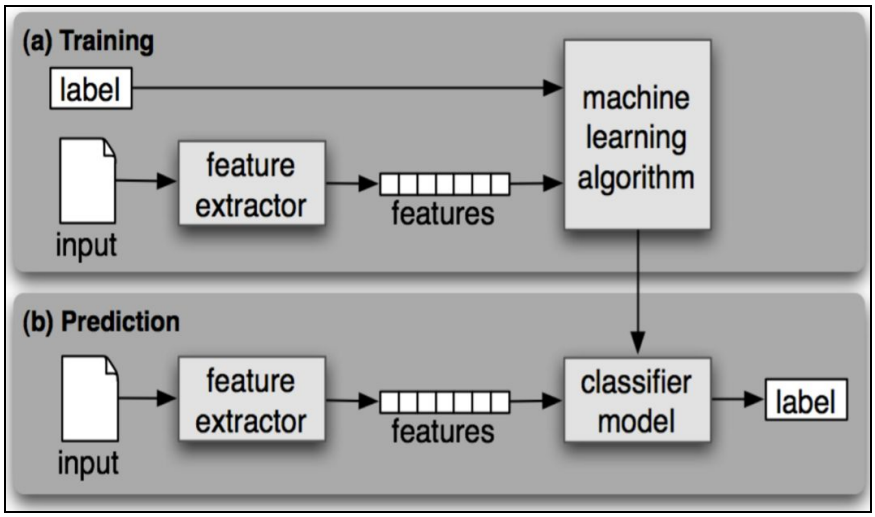
- Understanding audience sentiment from social media
- Detection of spam and non-spam emails
- Auto-tagging of customer queries
- Categorization of news articles into defined topics.

For this project, we have used a dataset from Kaggle which consists of about 1.7 M records. The data is already divided into train and test sets with the train set having around 1.3 M records and the test set 380K records.
The dataset has three fields:

1. **qid** - Unique question identifier
2. **question_text** - Quora question text
3. **target** - a question labeled “insincere” has a value 1, otherwise, 0

Our end-to-end Text Classification Project Approach

1. Dataset Preparation: The first step is the dataset preparation step which includes the process of loading a dataset and performing basic pre-processing. The dataset is then split into train and validation sets.
2. Feature Engineering: The next step is the feature engineering in which the raw dataset is transformed into flat features which can be used in a machine learning model. This step also includes the process of creating new features from the existing data.
3. Model Training: The final step is the model building step in which a machine learning model is trained on a labeled dataset.
4. Improve Performance of Text Classifier: In this article, we will also look at the different ways to improve the performance of text classifiers.



The above figure shows the general approach employed by us for the project. We started from the training set which had the labels for the insincere and sincere questions. Using this dataset consisting of labels as the input to our feature extraction algorithm, we found out the important features and characteristics of the dataset. We then trained these features obtained on various machine learning algorithms that we have discussed below. This provided us with the training accuracies and other parameters like F1-score, precision & recall for the training set.

The next step involved using the test data on the algorithms implemented to test for the prediction accuracy and the F1- score for this data and then check for the classification labels obtained for the questions present in the test set.

Analyzing our Data: Data Exploration

```
[10]  1 train_df = pd.read_csv("drive/My Drive/train.csv")
      2 given_test_df = pd.read_csv("drive/My Drive/test.csv")
      3 print("Train shape : ",train_df.shape)
      4 print("Test shape : ",given_test_df.shape) #useless

      Train shape :  (1306122, 3)
      Test shape :  (375806, 2)
```

To start with, we imported all the required libraries.

- Pandas (https://pandas.pydata.org/pandas-docs/stable/install.html)
- Scikit-learn (http://scikit-learn.org/stable/install.html)
- XGBoost (http://xgboost.readthedocs.io/en/latest/build.html)
- TextBlob (http://textblob.readthedocs.io/en/dev/install.html)
- Keras (https://keras.io/#installation)

Pandas is an open-source library providing high performance and easy to use data structures and data analysis tools. In this project, we used it for a number of data analysis tasks and also to create data frames.
Scikit-learn is an open-source library that is primarily used for data mining and employing machine learning approaches. Here, we used it for implementing the machine learning algorithms that we used on the test data.
XGBoost is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable. Here, we used it to implement the boosting trees as a part of our machine learning approach.
TextBlob is a library for processing textual data and it provides simple API for diving into common natural language processing tasks such as PoS tagging, sentiment analysis, classification and more.
Keras is a deep learning library in python and is capable of running on top of TenserFlow. This was used during the deep learning part of the project.

Dataset Preparation and Description

Below is the more detailed description (similar to described in the above section) of the data and the details about the data preparation methods applied

• **Total number of data points:** 1.7 M
• **Data Fields:**
    1. **qid** - unique question identifier
    2. **question_text** - Quora question text
    3. **target** - a question labeled "insincere" has a value of 1, otherwise 0
• **Training data points:** 1.3M
• **Test data points:** 376 K

• **Word Embeddings used:** GoogleNews-vectors-negative300 , glove.840B.300d, paragram_300_sl999 , wiki-news-300d-1M
• **Label/output to predict:** target (0 for sincere, 1 for insincere)
• **Data Cleaning :**
    1. Regex for removing special characters
    2. Missing values handled using filling NA's
    3. Tokenize the sentences
    4. Pad the sentences

Regular expression method was used to form generalized rules for the sentences that occur as questions in the text field with certain rules imposed on it. This helped to get rid of special characters and other unnecessary characters present in the sentences.

Missing values create a lot of ambiguity during classification. Hence, these values are replaced with certain global constant values and some rows where the class label is missing are ignored. In this way, missing attributes in the dataset are handled by cleaning the dataset.

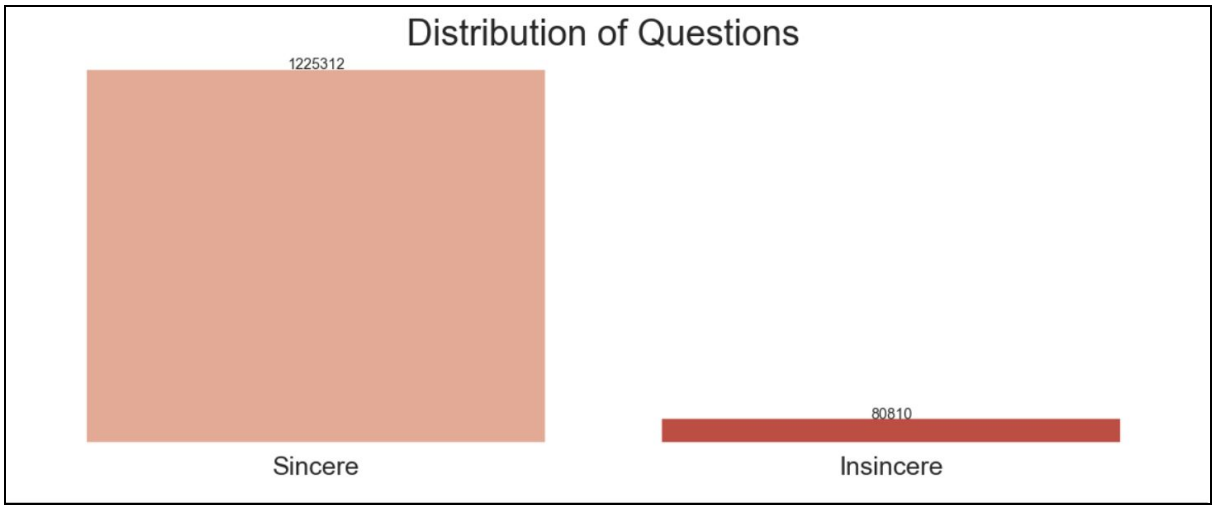
Once the missing values were handled and the sentences were obtained in a proper format after using regular expressions and lower casing them, tokenization was applied to get words as the tokens in a list format.

Padding: Strings can be padded with spaces or other characters to a certain width. For example, some numerical, abbreviated or textual data are often represented with prepending blank space to ensure the data is unclustered and unambiguous for classification.

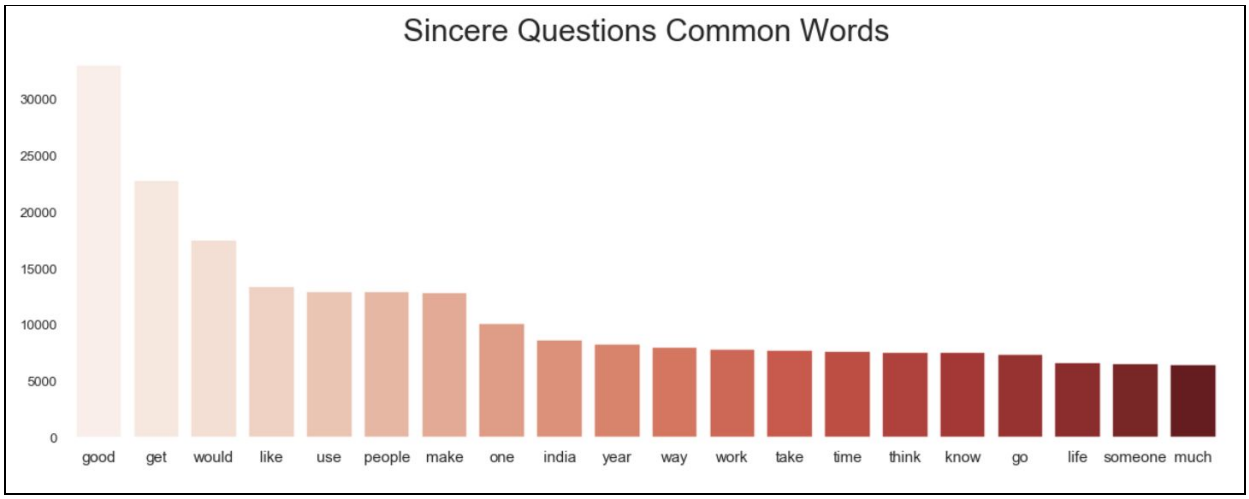
```
[ ]  1 train_df[train_df.target==1].head(20)
```

	qid	question_text	target
22	0000e91571b60c2fb487	Has the United States become the largest dictatorship in the world?	1
30	00013ceca3f624b09f42	Which babies are more sweeter to their parents? Dark skin babies or light skin babies?	1
110	0004a7fcb2bf73076489	If blacks support school choice and mandatory sentencing for criminals why don't they vote Republican?	1
114	00052793eaa287aff1e1	I am gay boy and I love my cousin (boy). He is sexy, but I dont know what to do. He is hot, and I want to see his di**. What should I do?	1
115	000537213b01fd77b58a	Which races have the smallest penis?	1
119	00056d45a1ce63856fc6	Why do females find penises ugly?	1
127	0005de07b07a17046e27	How do I marry an American woman for a Green Card? How much do they charge?	1
144	00068875d7c82a5bcf88	Why do Europeans say they're the superior race, when in fact it took them over 2,000 years until mid 19th century to surpass China's largest economy?	1
156	0006ffd99a6599ff35b3	Did Julius Caesar bring a tyrannosaurus rex on his campaigns to frighten the Celts into submission?	1
167	00075f7061837807c69f	In what manner has Republican backing of 'states rights' been hypocritical and what ways have they actually restricted the ability of states to make their own laws?	1
168	00076debbd82860ca33a	Would Europeans continue to participate in the Arab war for the destruction of Israel and killing all the Jews, if they knew that god himself defends Israel and he will do to Europeans what the Arabs want to do to the Jews?	1
208	000983f5b226cca636f4	Why are Americans, British, Canadians, Australians and New Zealanders considered to be separate nations even when they all speak the same language?	1
214	0009fcb845bb24de91f4	If both Honey Singh and Justin Bieber fall from the 5th floor, who will survive?	1
224	000a898565e80fe124bf	Why are liberal minorities so voilent towards poeple with diffrent poltical beleifs? Should supporting trump be a sentence to be imprisoned or savegely attacked?	1
235	000b1f4cbc5c238a765d	Can we all now admit that President Trump doesn't really want Congress to pass legislation replacing DACA to protect dreamers?	1
260	000c44fe9b2822b25d77	The American economy is growing under Trump's presidency, why do people still hate Trump as a president?	1
286	000d9de4df8d6a31c248	Why don't poor countries print more money to use for paying for education, etc.?	1
289	000db4d0223af5dd9c5d	Why do all the people who claim Florida has great weather go silent every time there's a new hurricane?	1
291	000db9603d16a6d9bb2e	Could the leader of Iran be dead many years ago and the leader of today's Iran is actually a fake leader?	1
302	000e67648fce55f011be	Why do the Liberals who run schools choose not to have controlled access? The kids in Florida were killed due to an unlocked door.	1

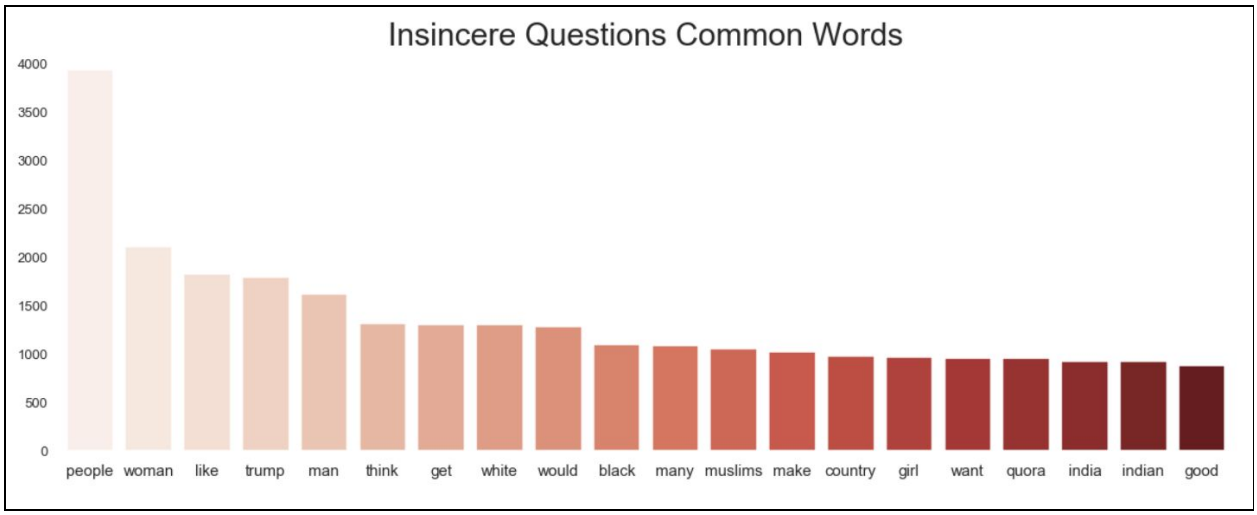
The above figure shows the distribution of data having insincere texts (labeled as 1). From the above questions, we get a general idea that the data categorized as insincere has a variety of sentences that do not make sense in any manner of speaking or are just plain statements.



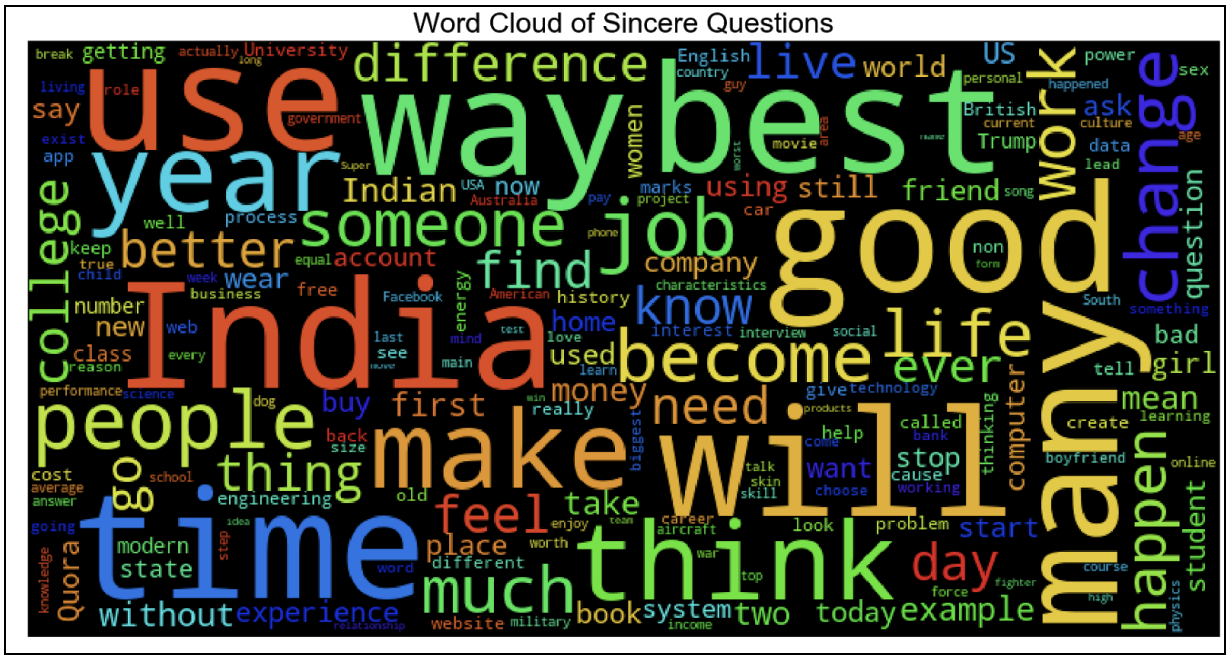
The above graph shows the distribution of questions labeled as insincere and sincere from the test data set. The above distribution shows that the data is highly undistributed with majority values labeled as sincere, i.e, with a 0 label.



The above graph shows the distribution of most commonly occurring words in the test data set that are classified as sincere. It can be seen that ‘good’ occurs in most of the questions with a count of above 3000 and ‘much’ occurs somewhere around 500 times.

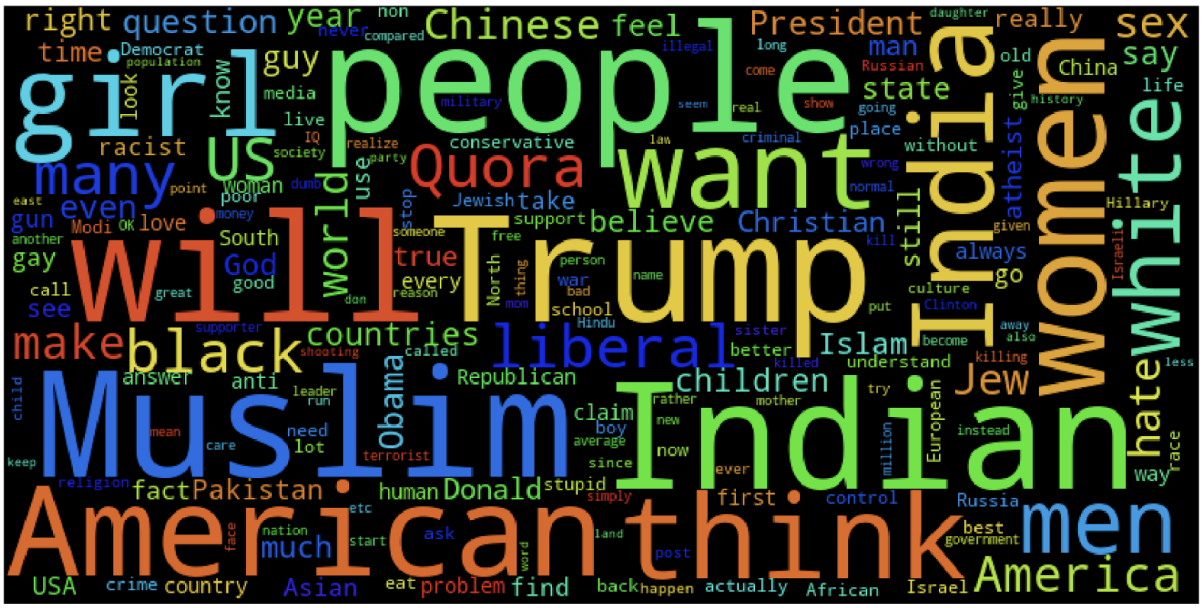


The above graph shows the distribution of most commonly occurring words in the test data set that are classified as insincere. It can be seen that ‘people’ occurs in most of the questions with a count of above 3500 and ‘good’ occurs somewhere around 1000 times.



The word cloud above consists of texts that appear most frequently in the sincere questions. The larger the size of the text, the more frequent the word in that category.

Word Cloud of Insincere Questions



The word cloud above consists of texts that appear most frequently in the insincere questions. The larger the size of the text, the more frequent the word in that category.

From the above word clouds, it can be seen that the frequency of certain similar words is more in both the categories, i.e. India and Indian occur quite frequently in both sincere and insincere questions present in the test data set. Also, there are some words in the insincere sentences word cloud such as American, Muslim, Trump and many others which do not make sense on its own.

Most common sincere n-grams

	word	freq	percentage
0	tips someone starting	716	12.402564
1	useful tips someone	713	12.350598
2	someone starting work	713	12.350598
3	advice give someone	640	11.086090
4	give someone moving	519	8.990126
5	good hotels short-term	519	8.990126
6	hotels short-term business	519	8.990126
7	short-term business travelers	519	8.990126
8	good bad neighborhoods	515	8.920838
9	best known for?	400	6.928807

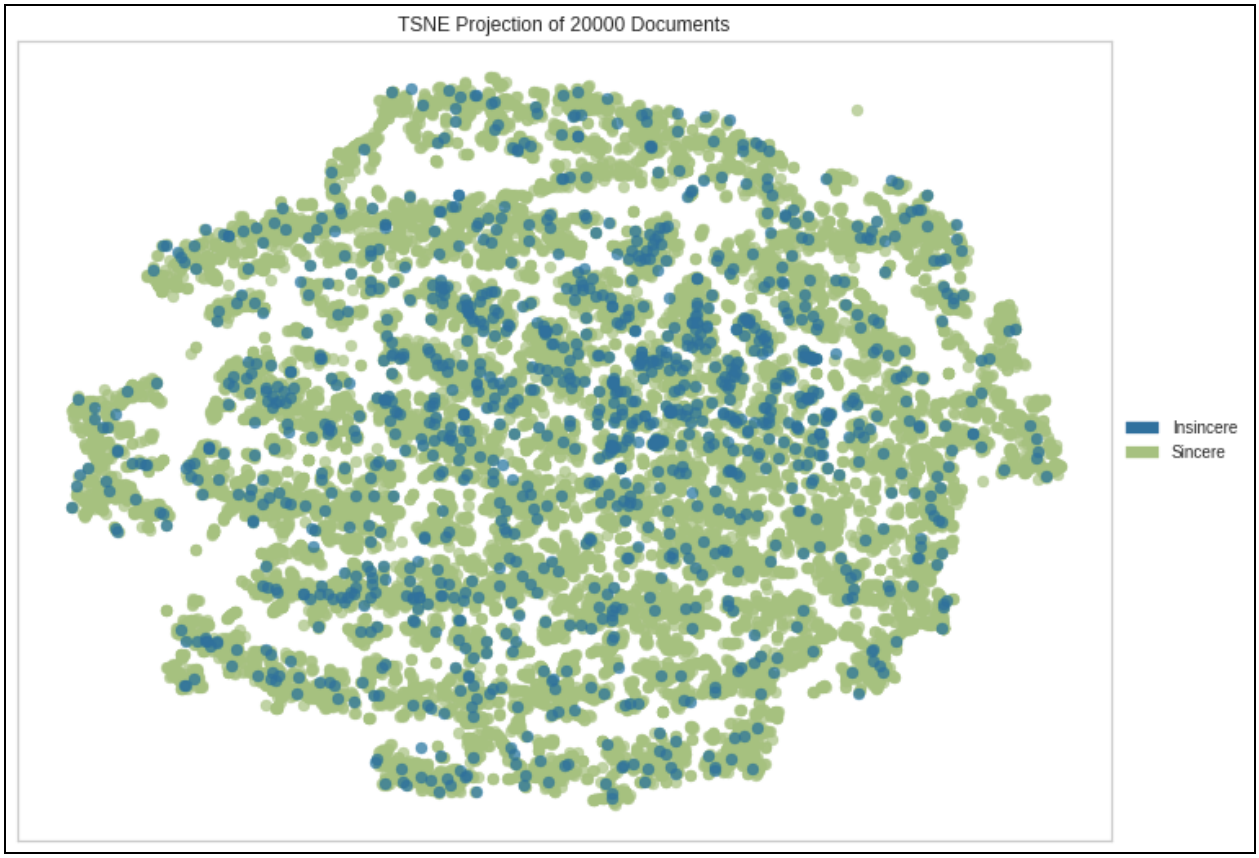
Most common insincere n-grams

	word	freq	percentage
0	will donald trump	43	12.250712
1	black lives matter	42	11.965812
2	long will take	38	10.826211
3	kim jong un	36	10.256410
4	12 year old	35	9.971510
5	14 year old	33	9.401709
6	people still believe	33	9.401709
7	united states america	31	8.831909
8	think donald trump	30	8.547009
9	ask stupid questions	30	8.547009

To observe a more clear distribution of words and to categorize them in any of the above two categories, we found out the most common sincere n-grams and most common insincere n-grams. The sentences obtained here as well do not make much sense either. They are random in both the cases and to identify and categorize them properly, it is necessary that appropriate machine learning algorithms are employed on this data so that most important characteristics from the data can be determined and the prediction models can be developed to be used on the test data.

t-SNE - Visual Cluster Plot

t-SNE (t-Distributed Stochastic Neighbor Embedding) is a technique for dimensionality reduction suited well for visualizing high-dimensional datasets



The data isn't very separable by traditional means. Hence a non-parametric model will most likely do the best (Gradient Boosting or Neural Networks).

The basic idea behind doing the above visualizations was to get to know how the data was distributed and what were the factors that categorized it into insincere and sincere sentences and observe those factors in a visual format.

Code Snippet:

Data pre processing

```
[12]
## split to train and val
train_df, val_df = train_test_split(train_df, test_size=0.2, rand
train_df, test_df = train_test_split(train_df, test_size=float(1.

## fill up the missing values
train_X = train_df["question_text"].fillna("_na_").values
val_X = val_df["question_text"].fillna("_na_").values
test_X = test_df["question_text"].fillna("_na_").values

## Tokenize the sentences
tokenizer = Tokenizer(num_words=max_features)
tokenizer.fit_on_texts(list(train_X))
train_X = tokenizer.texts_to_sequences(train_X)
val_X = tokenizer.texts_to_sequences(val_X)
test_X = tokenizer.texts_to_sequences(test_X)

## Pad the sentences
train_X = pad_sequences(train_X, maxlen=maxlen)
val_X = pad_sequences(val_X, maxlen=maxlen)
test_X = pad_sequences(test_X, maxlen=maxlen)

## Get the target values
train_y = train_df['target'].values
val_y = val_df['target'].values
test_y = test_df['target'].values
```

Feature Engineering

The next step is the feature engineering step. In this step, raw text data was transformed into feature vectors and new features will be created using the existing dataset. We implemented the following different ideas in order to obtain relevant features from our dataset.

TF-IDF Vectors as features

- **N-Gram level**: N-grams are the combination of N terms together. This Matrix representing TF-IDF scores of N-grams (represents the relative importance of a term in the document and the entire corpus)

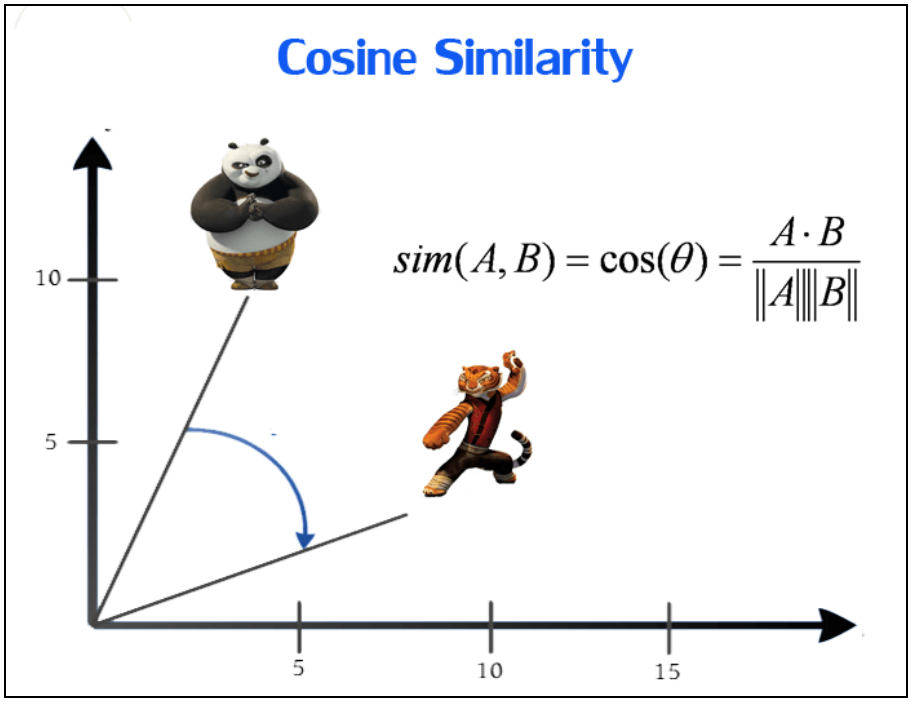
$$\text{TF}(\text{word}, \text{text}) = \frac{\text{number of times the word occurs in the text}}{\text{number of words in the text}}$$

$$\text{IDF}(\text{word}) = \textbf{log}\left[\frac{\text{number of texts}}{\text{number of texts where the word occurs}}\right]$$

$$\text{TF-IDF}(\text{word}, \text{text}) = \text{TF}(\text{word}, \text{text}) \times \text{IDF}(\text{word})$$

Word Embeddings as Features

- GoogleNews-vectors-negative300
- glove.840B.300d
- Paragram_300_sl999
- wiki-news-300d-1M



Idea: Distributed representations- Intuitively, there would be some dependence of one word on the other words. The words in the context of this word would get a greater share of this dependence. Having words with similar context occupy close spatial positions, mathematically, the cosine of the angle between such vectors should be close to 1, i.e. angle close to 0.

Word embedding: A form of representing words and documents using a dense vector representation. The position of a word within the vector space is learned from the text and is based on the words that surround the word when it is used. N-grams of words/characters represented as a vector (N-grams are overlapping groups of multiple succeeding words/characters in the text).
There are four essential steps:
1. Loading the pre-trained word embeddings
2. Creating a tokenizer object

3. Transforming text documents into a sequence of tokens and padding them
4. Create a mapping of token and their respective embeddings

GloVe is an unsupervised learning algorithm for obtaining vector representations for words. Training is performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space. Word2vec is a particularly computationally-efficient predictive model for learning word embeddings from raw text. 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. This inversion might seem like an arbitrary choice, but statistically, it has the effect that CBOW smoothes over a lot of the distributional information (by treating an entire context as one observation). For the most part, this turns out to be a useful thing for smaller datasets.

Code snippet:

Word2vec Embeddings:

```
from gensim.models import KeyedVectors

EMBEDDING_FILE = 'drive/My Drive/embeddings/GoogleNews-vectors-ne
embeddings_index = KeyedVectors.load_word2vec_format(EMBEDDING_FI

word_index = tokenizer.word_index
nb_words = min(max_features, len(word_index))
embedding_matrix_4 = (np.random.rand(nb_words, embed_size) - 0.5)
for word, i in word_index.items():
    if i >= max_features: continue
    if word in embeddings_index:
        embedding_vector = embeddings_index.get_vector(word)
        embedding_matrix_4[i] = embedding_vector
```

Combined word embeddings code snippet:

Combine :

```
[17]  embedding_matrix = np.concatenate((embedding_matrix_1, embedding_
      np.shape(embedding_matrix)
```

```
(50000, 1200)
```

Major NLP techniques reported in the researched resources

We split the dataset into training, validation, and testing in the 70%, 20%, and 10%
and evaluated results based on F1 Accuracy. For improving Text Classification Models

- **Text Cleaning:** text cleaning helped to reduce the noise present in text data in the form of stopwords, punctuations marks, suffix variations, etc. using Regex, filling NA's. Better pre-processing of input data can be done as per the need of your dataset like removing some special symbols, numbers, stopwords and so on …

Data pre processing

```
[ ]
import re
def clean_text(x):

    x = str(x)
    x = re.sub('[0-9]{5,}', '#####', x)
    x = re.sub('[0-9]{4}', '####', x)
    x = re.sub('[0-9]{3}', '###', x)
    x = re.sub('[0-9]{2}', '##', x)
    for punct in '?!.':
        x = x.replace(punct, ' EOS ')
    for punct in "/-'":
        x = x.replace(punct, ' ')
    for punct in '&':
        x = x.replace(punct, ' ' + punct+' ')
    for punct in ',"#$%\'()*+-/:;<=>@[\\]^_`{|}~' + '“”’':
        x = x.replace(punct, '')

    return x
```

- 2. **Stacking Text / NLP features with text feature vectors:** In the feature engineering section, we used a number of different feature vectors, combining them together can help to improve the accuracy of the classifier.
- 3. **Hyperparameter Tuning in modeling:** Tuning the parameters is an important step, a number of parameters such as tree length, network parameters, etc were fine-tuned to get the best fit model.
- 4. **K-Fold Cross-Validation** was used as a resampling procedure used to evaluate machine learning models on a limited data sample. It generally results in a less biased or less optimistic estimate of the model skill than other methods, such as a simple train/test split.
- 5. **Use Dropout Layer:** Dropout is a regularization technique to avoid overfitting (increase the validation accuracy) thus increasing the generalizing power.

Model Building - Classifier Models & Comparison Metrics

The final step in the text classification framework is to train a classifier using the features created in the previous step. There are many different choices of machine learning models which can be used to train a final model. We will implement the following different classifiers for this purpose

Logistic Regression

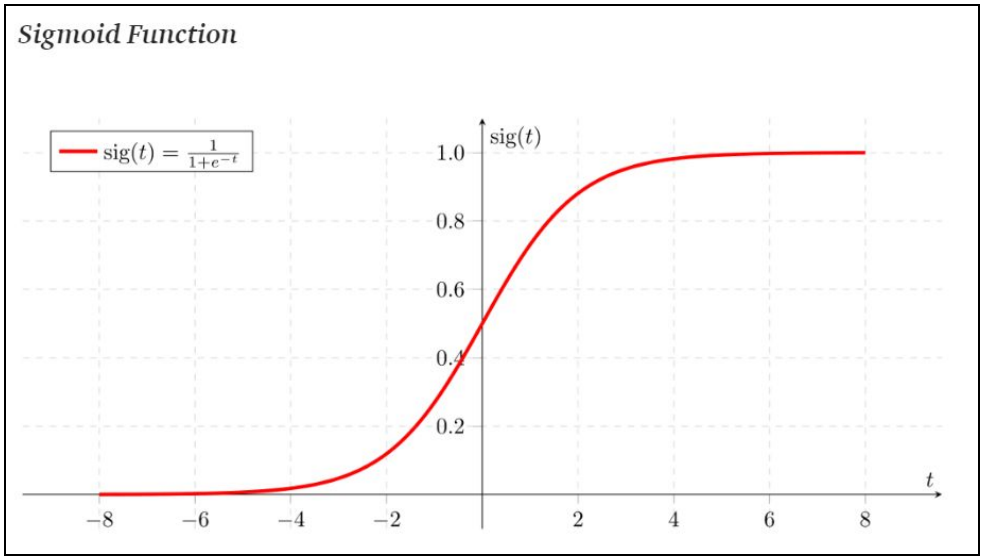
Logistic regression measures the relationship between the categorical dependent variable and one or more independent variables by estimating probabilities using a logistic/sigmoid function. Logistic Regression is used when the dependent variable(target) is categorical. Since our project consists prediction of a target variable in a particular category whether it is a sincere question or an insincere quora question we have implemented the logistic based regression as the base algorithm for classification using the machine learning approach.

Model

Output = 0 or 1

Hypothesis => Z = WX + B

hΘ(x) = sigmoid (Z)



If ‘Z’ goes to infinity, Y(predicted) will become 1 and if ‘Z’ goes to negative infinity, Y(predicted) will become 0.

This justifies the name 'logistic regression'. Data is fit into a linear regression model, which then be acted upon by a logistic function predicting the target categorical dependent variable.
Binary logistic regression major assumptions:

1. The dependent variable should be dichotomous in nature (e.g., presence vs. absent).
2. There should be no outliers in the data, which can be assessed by converting the continuous predictors to standardized scores, and removing values below -3.29 or greater than 3.29.
3. There should be no high correlations (multicollinearity) among the predictors. This can be assessed by a correlation matrix among the predictors. Tabachnick and Fidell (2013) suggest that as long correlation coefficients among independent variables are less than 0.90 the assumption is met.

At the center of the logistic regression analysis is the task of estimating the log odds of an event. Mathematically, logistic regression estimates a multiple linear regression function defined as:

logit(p) $= \log\left(\frac{p(y=1)}{1-(p=1)}\right) = \beta_0 + \beta_1 x_{i2} + \beta_2 \cdot x_{i2} + \_ + \beta_p \cdot x_{in}$ for i = 1…n

Code snippet (Train):

```
accuracy_score(train_target, pred_train)
```

```
0.9491532950214452
```

```
pred_train = (oof_preds > .25).astype(np.int)
f1_score(train_target, pred_train)
```

```
0.6150028985507247
```

Code snippet(Validation):

```
accuracy_score(valid_target, pred_valid)
```

```
0.9468503768919175
```

```
pred_train = (oof_preds > .25).astype(np.int)
f1_score(valid_target, pred_valid)
```

```
0.6069465277609237
```

Code snippet(Test):

```
accuracy_score(test_target, pred_test)
```

```
0.9437639188295318
```

```
pred_train = (oof_preds > .25).astype(np.int)
f1_score(test_target, pred_test)
```

```
0.6085872940924723
```

Naive Bayes Classifier

Naive Bayes is a classification technique based on Bayes' Theorem with an assumption of independence among predictors. A Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature. Naive Bayes is popular for text classification.

The main goal of Naive Bayes Classifier is to determine the probabilities of feature occurring in each class and to return the class with the highest probability. This can be done with the help of the Bayes rule.

The formula for Bayes rule is given as: $$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

But to use in classification problem we have to use a slightly different representation of the above equation. The formula which we use for classification is:

$$P(c_i|x_0,\ldots,x_n) \propto P(x_0,\ldots,x_n|c_i)P(c_i)$$
$$\propto P(c_i)\prod_{j=1}^{n} P(x_j|c_i)$$

In our project, we have used Multinomial Naive Bayes classifier for text classification. We used this type of Naive Bayes classifier as it multinomial classifier is best suited for text classification.

Code snippet (Train):

```
accuracy_score(train_target, pred_train)
```

```
0.9223188951721202
```

```
pred_train = (oof_preds1 > .25).astype(np.int)
f1_score(train_target, pred_train)
```

```
0.5322459441890914
```

Code snippet(Validation):

```
accuracy_score(valid_target, pred_valid)
```

```
0.9194495609434598
```

```
pred_train = (oof_preds1 > .25).astype(np.int)
f1_score(valid_target, pred_valid)
```

```
0.5287439914092398
```

Code snippet(Test):

```
pred_train = (oof_preds1 > .25).astype(np.int)
f1_score(test_target, pred_test)
```

```
0.5292944062551329
```

```
accuracy_score(test_target, pred_test)
```

```
0.9173284522740934
```

Random Forest

Random Forest models are a type of ensemble models, particularly bagging models. They are part of the tree based model family.
Rather than simply averaging the prediction of the decision trees, the model uses two key concepts that give it the name random

1. Random sampling of training data points when building trees
2. Random subsets of features considered when splitting nodes

The Random Forest method introduces more randomness and diversity by applying the bagging method to the feature space. That is, instead of searching greedily for the best predictors to create branches, it randomly samples elements of the predictor space, thus adding more diversity and reducing the variance of the trees at the cost of equal or higher bias. This process is also known as “feature bagging” and it is this powerful method what leads to a more robust model.

Code snippet (Train):

```
accuracy_score(train_target, pred_train)
```

```
0.9512439539203487
```

```
pred_train = (oof_preds > .25).astype(np.int)
f1_score(train_target, pred_train)
```

```
0.6510239259247383
```

Code snippet(Validation):

```
accuracy_score(valid_target, pred_valid)
```

```
0.9501258397753683
```

```
pred_train = (oof_preds > .25).astype(np.int)
f1_score(valid_target, pred_valid)
```

```
0.6473287419248734
```

Code snippet(Test):

```
accuracy_score(test_target, pred_test)
```

```
0.9467345925873382
```

```
pred_train = (oof_preds > .25).astype(np.int)
f1_score(test_target, pred_test)
```

```
0.6437936912765623
```

XGBoost

Boosting models are another type of ensemble models part of tree-based models. Boosting is a machine learning ensemble meta-algorithm for primarily reducing bias, and also variance in supervised learning, and a family of machine learning algorithms that convert weak learners to strong ones. A weak learner is defined to be a classifier that is only slightly correlated with the true classification (it can label examples better than random guessing).

It is a highly flexible and versatile tool that can work through most regression, classification and ranking problems as well as user-built objective functions.ts name stands for **eXtreme Gradient Boosting**, it was developed by Tianqi Chen and now is part of a wider collection of open-source

libraries developed by the Distributed Machine Learning Community (DMLC). XGBoost is a scalable and accurate implementation of gradient boosting machines and it has proven to push the limits of computing power for boosted trees algorithms as it was built and developed for the sole purpose of model performance and computational speed.

Code snippet (Train):

```
accuracy_score(train_target, pred_train)
```

```
0.9449875279644627
```

```
pred_train = (oof_preds > .25).astype(np.int)
f1_score(train_target, pred_train)
```

```
0.47360825195419815
```

Code snippet(Validation):

```
accuracy_score(valid_target, pred_valid)
```

```
0.9439349875326621
```

```
pred_train = (oof_preds > .25).astype(np.int)
f1_score(valid_target, pred_valid)
```

```
0.4729340933845974
```

Code snippet(Test):

```
accuracy_score(test_target, pred_test)
```

```
0.9423583498273929
```

```
pred_train = (oof_preds > .25).astype(np.int)
f1_score(test_target, pred_test)
```

```
0.4716459834671187
```

SVM

A Support Vector Machine (SVM) is a supervised machine learning algorithm that can be employed for both classification and regression purposes. It is based on the idea of finding a hyperplane that best divides a dataset into two classes.

SVM is used for text classification tasks such as category assignment, detecting spam and sentiment analysis. It is also commonly used for image recognition challenges, performing particularly well in aspect-based recognition and color-based classification. It also plays a vital role in many areas of handwritten digit recognition, such as postal automation services.

Code snippet (Train):

```
accuracy_score(train_target, pred_train)
```

```
0.9323873492842364
```

```
pred_train = (oof_preds > .25).astype(np.int)
f1_score(train_target, pred_train)
```

```
0.5245736574773576
```

Code snippet(Validation):

```
accuracy_score(valid_target, pred_valid)
```

```
0.9327653874863469
```

```
pred_train = (oof_preds > .25).astype(np.int)
f1_score(valid_target, pred_valid)
```

```
0.5278646846953756
```

Code snippet(Test):

```
accuracy_score(test_target, pred_test)
```

```
0.9324233283023940
```

```
pred_train = (oof_preds > .25).astype(np.int)
f1_score(test_target, pred_test)
```

```
0.5245362423653455
```

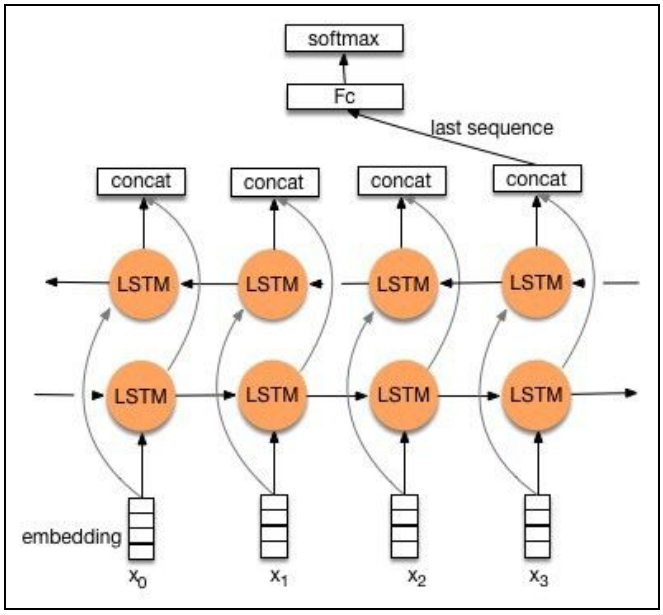
Deep Neural Networks

Deep Neural Networks are more complex neural networks in which the hidden layers perform much more complex operations than simple sigmoid or relu activations. Different types of deep learning models were applied in this text classification problem

Bidirectional LSTM:

Unlike Feed-forward neural networks wherein activation outputs are propagated only in one direction, in bidirectional LSTM, the activation outputs from neurons propagate in both directions (from inputs to outputs and from outputs to inputs). This creates loops in the neural network

architecture which acts as a ‘memory state’ of the neurons. This state allows the neurons an ability to remember what has been learned so far. The memory state gives an advantage over traditional neural networks but a problem called Vanishing Gradient is associated with them. In this problem, while learning with a large number of layers, it becomes really hard for the network to learn and tune the parameters of the earlier layers. To address this problem, a new type of RNNs called LSTMs (Long Short Term Memory) Model has been utilized. RNN layers were wrapped in Bidirectional layers here.



Code snippet (Train):

```
11
12 # Fit the model
13 model_lstm.fit(train_X, train_y, validation_split= 0.2, epochs=2, batch_size=256)
```

```
Train on 731427 samples, validate on 182857 samples
Epoch 1/2
731427/731427 [==============================] - 71s 98us/step - loss: 0.1570 - acc: 0.9509 - val_loss: 0.1150 - val_acc: 0.9549
Epoch 2/2
731427/731427 [==============================] - 70s 96us/step - loss: 0.1128 - acc: 0.9571 - val_loss: 0.1137 - val_acc: 0.9551
<keras.callbacks.History at 0x7ff94eb990b8>
```

Code snippet (Validation):

```
[42]  1 #Calculate F1 score on validation data
      2 f1_quora  = f1_score(val_y, valid_pred_01_lstm)
      3 print('Validation F1 score LSTM:', f1_quora)
```

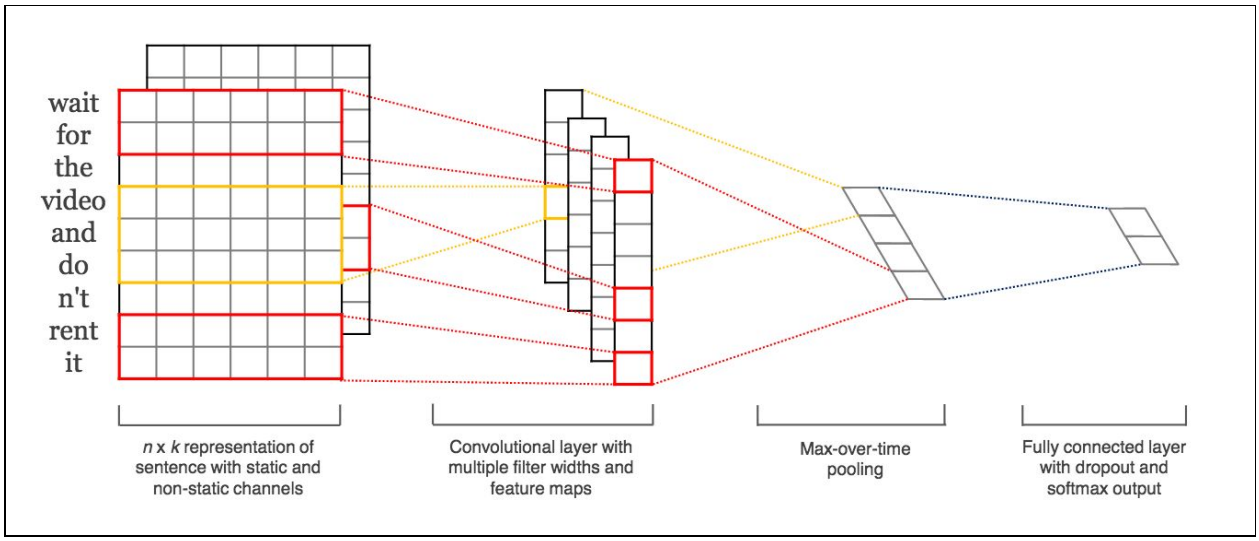
```
Validation F1 score LSTM: 0.6397645108742248
```

```
[49]  1 submit_df.prediction.value_counts()

      0    121292
      1      9321
      Name: prediction, dtype: int64
```

Convolutional Neural Network (CNN)

In convolutional neural networks, convolutions over the input layer are used to compute the output. This results in local connections, where each region of the input is connected to a neuron in the output. Each layer applies different filters and combines their results.



Code snippet (Train):

```
[24]  1 model_cnn.fit(train_X, train_y, batch_size=512, epochs=2, validation_data=(val_X, val_y))

      WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow/python/ops/math_ops.py:3066: to_int32 (from tensorflow.py
      Instructions for updating:
      Use tf.cast instead.
      Train on 914284 samples, validate on 261225 samples
      Epoch 1/2
      914284/914284 [==============================] - 472s 516us/step - loss: 0.1232 - acc: 0.9530 - val_loss: 0.1048 - val_acc: 0.9583
      Epoch 2/2
      914284/914284 [==============================] - 462s 506us/step - loss: 0.1005 - acc: 0.9601 - val_loss: 0.1046 - val_acc: 0.9585
```

Code snippet (Validation):

```
[29]
#Validation
pred_val_y =  pred_val_cnn_y

thresholds = []
for thresh in np.arange(0.1, 0.501, 0.01):
    thresh = np.round(thresh, 2)
    res = metrics.f1_score(val_y, (pred_val_y > thresh).astype(int))
    thresholds.append([thresh, res])
    print("F1 score at threshold {0} is {1}".format(thresh, res)) #validation

thresholds.sort(key=lambda x: x[1], reverse=True)
best_thresh = thresholds[0][0]
print("Best threshold: ", best_thresh)
```

```
F1 score at threshold 0.1 is 0.6176863478493188
F1 score at threshold 0.11 is 0.6259294740203458
F1 score at threshold 0.12 is 0.6332012124038238
F1 score at threshold 0.13 is 0.6388208724592892
F1 score at threshold 0.14 is 0.6430317256733539
F1 score at threshold 0.15 is 0.6467372568332924
F1 score at threshold 0.16 is 0.6503251625812906
F1 score at threshold 0.17 is 0.6533556876492406
F1 score at threshold 0.18 is 0.6556870022423258
F1 score at threshold 0.19 is 0.6574991503490104
F1 score at threshold 0.2 is 0.6592655397170261
F1 score at threshold 0.21 is 0.6602056431045128
F1 score at threshold 0.22 is 0.6618090862129904
F1 score at threshold 0.23 is 0.6621710888472971
F1 score at threshold 0.24 is 0.6628428233328693
F1 score at threshold 0.25 is 0.6632127270167535
F1 score at threshold 0.26 is 0.6627010103885014
F1 score at threshold 0.27 is 0.6631972397929844
F1 score at threshold 0.28 is 0.663200999041617
F1 score at threshold 0.29 is 0.6627944498225233
F1 score at threshold 0.3 is 0.6625977719838824
F1 score at threshold 0.31 is 0.6615416828691932
F1 score at threshold 0.32 is 0.6612318840579711
F1 score at threshold 0.33 is 0.659733447192656
F1 score at threshold 0.34 is 0.6577974870657798
F1 score at threshold 0.35 is 0.6559855649576903
F1 score at threshold 0.36 is 0.654284190380573
F1 score at threshold 0.37 is 0.652183551318998
F1 score at threshold 0.38 is 0.650223642172524
F1 score at threshold 0.39 is 0.6482251668439888
F1 score at threshold 0.4 is 0.646808233878573
F1 score at threshold 0.41 is 0.6449011702232275
F1 score at threshold 0.42 is 0.6429326986645512
F1 score at threshold 0.43 is 0.6396666666666666
F1 score at threshold 0.44 is 0.63671796250252
F1 score at threshold 0.45 is 0.6335681387157951
F1 score at threshold 0.46 is 0.6318484383000512
F1 score at threshold 0.47 is 0.6284356271957016
F1 score at threshold 0.48 is 0.625615763546798
F1 score at threshold 0.49 is 0.621225375275552
F1 score at threshold 0.5 is 0.6175775323713085
Best threshold:  0.25
```

Code snippet (Test):

```
[50]
# Test data predictions
test_y = test_df['target'].values
pred_test_y =  pred_test_cnn_y

thresh = .25
res = metrics.f1_score(test_y, (pred_test_y > thresh).astype(int))
print("F1 score_test at threshold {0} is {1}".format(thresh, res)) #test

F1 score_test at threshold 0.25 is 0.6592970521541951

[51]
# Test accuracy CNN
# evaluate the model
  scores = model_cnn.evaluate(test_X, test_y, verbose=0)
  print("%s: %.2f%%" % (model_cnn.metrics_names[1], scores[1]*100))

acc: 95.82%

[52]
pred_test_y = (pred_test_y > best_thresh).astype(int)
out_df = pd.DataFrame({"qid":test_df["qid"].values})
out_df['prediction'] = pred_test_y
out_df.to_csv("drive/My Drive/submission_Apr29.csv", index=False)

out_df.head()
```

	qid	prediction
0	6b6bb83cc6e7a3eabef2	0
1	ad74a0acab984c050fa8	0
2	164437e082b6fc7c63ab	1
3	5cdf1a0477b440c53a76	0
4	7fd964b013043abb8791	0

Results - Prediction Performance

Comparing the different types of Classification Models

	Classifier	Accuracy			F-1 Score		
		Train	Val	Test	Train	Val	Test
1	**Logistic Regression**	0.9491	0.946	0.943	0.615	0.606	0.608
2	**Naive Bayes**	0.922	0.919	0.917	0.532	0.528	0.529
3	**Random Forest**	0.951	0.950	0.946	0.651	0.647	0.643
4	**XGBoost**	0.944	0.943	0.942	0.473	0.472	0.471
5	**SVM**	0.932	0.932	0.933	0.524	0.527	0.524
6	**Bidirectional LSTM**	0.950	0.955	0.954	0.691	0.641	0.620
7	**Convolutional Neural Network (CNN)**	0.960	0.959	0.954	0.709	0.663	0.659

Best Classification Models: Detailed Performance:

	Classifier	Accuracy			F-1 Score		
		Train	Val	Test	Train	Val	Test
1	Bidirectional LSTM	0.9501	0.9551	.9549	0.6915	0.6410	0.6202
2	Convolutional Neural Network (CNN)	0.9600	0.9585	.9545	0.7091	0.6638	0.6593

Results- Inference and Discussion

- We observe that insincere questions contain highly debated and emotion based topics of Trump, Muslims, America, Russia, Obama, Liberal, etc. We observe the pattern of suggestive/controversial n-grams amongst the insincere questions.
- There were also few cases wherein a few words were found to be common between both sincere and insincere buckets. As we can see from the above table, simple BOW classification does not work with Logistic and Naive Bayes algorithms as these just count the frequency of words.
- Hence needed to go beyond it for deeper meaningful analysis at the semantic level. The traditional algorithms are also not adapt for imbalanced classes scenario such as ours.
- What slightly works better is XG Boost since it is more capable of scenario-based classification and hence overcomes the drawbacks of logistic and Naive Bayes. We can see here that the data isn't very separable by traditional means. A nonparametric model will most likely do the best (Gradient Boosting or Neural Networks).
- Deep learning algorithms such as Bidirectional LSTM and CNN seem to outperform the results of classical classifiers in this scenario and also require the minimum effort in data pre-processing.

Conclusion and Discussion

Overall, our models for predicting sincere and insincere questions performed well.

- The traditional machine learning algorithm had some positives and cons for classification due to the imbalanced data set.
- Among the traditional machine learning algorithm used Random Forest Algorithm gave us the best accuracy for classification since it uses the decision tree approach for selecting the most important feature and performs well for any classification problem.
- The CNN model closely followed by the Bidirectional LSTM performed the best.
- This was likely due to their unique ability to capture complex feature dependencies and complex interactions.
- Our model can be used by Quora in determining where to flag a question as insincere and automatically filter out them for revision by the author.

References

- Deshpande, A. (n.d.). A Beginners Guide To Understanding Convolutional Neural Networks Part 2. Retrieved from https://adeshpande3.github.io/adeshpande3.github.io/A-Beginners-Guide-To-Understanding-Convolutional-Neural-Networks-Part-2

- What is the Difference Between a Batch and an Epoch in a Neural Network? (2018, April 24). Retrieved from https://machinelearningmastery.com/difference-between-a-batch-and-an-epoch/
- Rohrer, B. (2017, June 27). Recurrent Neural Networks (RNN) and Long Short-Term Memory (LSTM). Retrieved from https://www.youtube.com/watch?v=WCUNPb-5EYI
- Deeplizard. (2017, December 09). Convolutional Neural Networks (CNN) explained. Retrieved from https://www.youtube.com/watch?v=YRhxdVk_sIs
- What is Logistic Regression? (n.d.). Retrieved from https://www.statisticssolutions.com/what-is-logistic-regression/
- Soni, D., & Soni, D. (2018, May 16). Introduction to Naive Bayes Classification. Retrieved from https://towardsdatascience.com/introduction-to-naive-bayes-classification-4cffabb1ae54
- Quora Insincere Questions Classification. (n.d.). Retrieved from https://www.kaggle.com/c/quora-insincere-questions-classification/data
- KDnuggets. (n.d.). Retrieved from https://www.kdnuggets.com/2017/10/random-forests-explained.html
- KDnuggets. (n.d.). Retrieved from https://www.kdnuggets.com/2017/10/xgboost-top-machine-learning-method-kaggle-explained.html
- KDnuggets. (n.d.). Retrieved from https://www.kdnuggets.com/2016/07/support-vector-machines-simple-explanation.html
- Simplified Text Processing. (n.d.). Retrieved from https://textblob.readthedocs.io/en/dev/
- Python Data Analysis Library. (n.d.). Retrieved from https://pandas.pydata.org/
- Learn. (n.d.). Retrieved from https://scikit-learn.org/stable/
- XGBoost Documentation. (n.d.). Retrieved from https://xgboost.readthedocs.io/en/latest/
- Keras: The Python Deep Learning library. (n.d.). Retrieved from https://keras.io/