Quora Insincere Text Classification

CS5100: Foundations of Artificial Intelligence Project Report

Camellia Debnath
MS in Data Science
Northeastern University

Viral Pandey
MS in Data Science
Northeastern University

Abstract

Quora [1] is a public question answer forum where anyone with an account can post questions or answers related to various topics, such as science, religion, politics, sports etc. With such a wide range of topics being discussed without any sort of regulation, quite often we observe questions or answers with insincere tones to them, i.e., questions or answers that are not neutral in nature, contains direct or indirect discrimination against some demography, or absurdity that is not grounded in reality. In this project, we aimed to build a binary classification model that successfully classifies Quora Texts into "sincere" and "insincere". We worked with a dataset provided by Quora on Kaggle [2], and fit various supervised and unsupervised machine learning classification models on the dataset. We compared the models on their ability to predict "sincere" vs "insincere" text on unseen data, and found out that Long Short Term Memory Recurrent Neural Networks (LSTM RNNs) gives the best results.

I. Introduction

The purpose of this project was to build a classification model to classify question text from Quora into two categories, namely "sincere" and "insincere". The data used for this purpose was taken from the Kaggle Competition titled "Quora Insincere Questions Classification" [2]. As a first step, we carried out sentiment analysis on the text to gauge the kind of words, or bi-grams, or tri-grams that we observe in the "sincere" questions vis-a-vis "insincere" questions, next we split the data into training and test sets. We preprocessed the training set to handle data imbalance, then we fit various supervised classification models to the data, such as: Logistic Regression, Support Vector Machines, Multi-Layer Perceptrons, Convolutional Neural Networks (CNNs), Long-Short Term Memory Recurrent Neural Networks (LSTM RNNs). We also fit an unsupervised machine learning model, namely k-means clustering. In the end, we compared the performance of the various classification models.

A. **Dataset**:

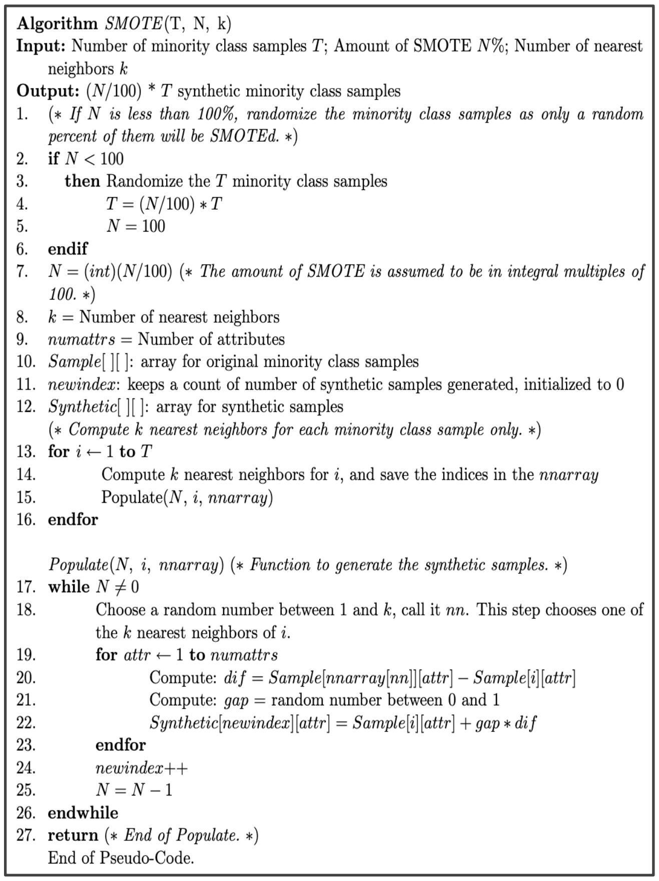
The training data contains the question text, and the "target" corresponding to teach text, indicating whether it was identified as sincere (target = 0) or insincere (target = 1).

II. Background

This paper explains the use of Synthetic Minority Oversampling Technique to handle class imbalance and evaluates a variety of different supervised machine learning binary classification algorithms and neural network architectures and optimization techniques as they relate to the task of classifying text data into binary class. The relevant existing techniques are explained in detail in the following sections.

A. Synthetic Minority Oversampling Technique (SMOTE):

Synthetic Minority Oversampling Technique is a method to create synthetic data points to overcome the problem of class imbalance. The minority class is over-sampled by taking each minority class sample and introducing synthetic examples along the line segments joining any/all of the k minority class nearest neighbors. Depending upon the amount of over-sampling required, neighbors from the k nearest neighbors are randomly chosen [3].



```
Algorithm SMOTE(T, N, k)
Input: Number of minority class samples $T$; Amount of SMOTE $N$%; Number of nearest
    neighbors $k$
Output: $(N/100) * T$ synthetic minority class samples
1.  (* If $N$ is less than 100%, randomize the minority class samples as only a random
    percent of them will be SMOTEd. *)
2.  if $N < 100$
3.     then Randomize the $T$ minority class samples
4.          $T = (N/100) * T$
5.          $N = 100$
6.  endif
7.  $N = (int)(N/100)$ (* The amount of SMOTE is assumed to be in integral multiples of
    100. *)
8.  $k$ = Number of nearest neighbors
9.  numattrs = Number of attributes
10. Sample[ ][ ]: array for original minority class samples
11. newindex: keeps a count of number of synthetic samples generated, initialized to 0
12. Synthetic[ ][ ]: array for synthetic samples
    (* Compute k nearest neighbors for each minority class sample only. *)
13. for $i \leftarrow 1$ to $T$
14.         Compute $k$ nearest neighbors for $i$, and save the indices in the nnarray
15.         Populate($N$, $i$, nnarray)
16. endfor

    Populate($N$, $i$, nnarray) (* Function to generate the synthetic samples. *)
17. while $N \neq 0$
18.        Choose a random number between 1 and $k$, call it nn. This step chooses one of
           the $k$ nearest neighbors of $i$.
19.        for attr $\leftarrow 1$ to numattrs
20.                Compute: $dif = Sample[nnarray[nn]][attr] - Sample[i][attr]$
21.                Compute: gap = random number between 0 and 1
22.                $Synthetic[newindex][attr] = Sample[i][attr] + gap * dif$
23.        endfor
24.        newindex++
25.        $N = N - 1$
26. endwhile
27. return (* End of Populate. *)
    End of Pseudo-Code.
```

SMOTE Pseudocode

B. Logistic Regression:

Logistic Regression is a very popular supervised machine learning classification method. The elegance of Logistic Regression lies in the simplicity of its implementation and interpretability. The logistic regression model can be represented as follows [4]:

$$\mathbf{y} = \boldsymbol{\sigma}(\mathbf{w}^{\mathbf{T}}\mathbf{x})$$

Where $\boldsymbol{\sigma}(.)$ is the logistic sigmoid function. Here y is the probability of a class as given by this equation, and **w** and **x** are the weight and feature vectors respectively.
We used the LogisticRegression function of Python scikit-learn library to implement this.

C. Support Vector Machine:

Support Vector Machines are supervised machine learning classification techniques, which are good for separating classes which are not linearly separable. The idea behind this classification technique is to find the maximum margin hyperplane (or line, in case of a 2-dimensional plane) in the p-dimensional feature space which separates the classes. Support Vector Machines use different Kernel Functions, which projects the non-linear non-separable input space into a higher dimensional linear separable space.

We used the SVC function of the Python scikit-learn library to fit SVM models to our oversampled training data using linear and kernel.

D. Neural Networks

Neural Networks are a class of supervised learning algorithm that learns a function mapping by training on given data. It learns by minimizing the objective loss function

$$L(W) = -\sum_{i=1}^{N}[Y_i log \pi_i + (1 - Y_i) log(1 - \pi_i)]$$

Here, W is the weight matrix and

$$\pi = \frac{1}{1 + e^{-(w_0 + w_1 x_1 + w2 x_2 .. + w_p x_p)}}$$

The weights are updated by calculating the predicted labels and then calculating the errors. The error is then used to calculate the gradients which are used to update the weights by backpropagation [4]. We use MLPClassifier of Scikit-Learn for fitting a simple feed-forward Neural Network model

E. Convolutional Neural Network:

Convolutional neural networks are deep artificial neural networks that are used primarily to classify images, cluster them by similarity, and perform object recognition within scenes. They are algorithms that can identify faces, individuals, street signs, tumors, platypuses and many other aspects of visual data [5]. Convolutional Neural Networks have a different architecture than regular Neural Networks. Convolution is one of the main building blocks of a CNN. The term convolution refers to the mathematical combination of two functions to produce a third function. It merges two sets of information. In the case of a CNN, the convolution is performed on the input data with the use of a filter or kernel to then produce a feature map. Convolution is performed by sliding the filter over the input. At every location, a matrix multiplication is performed and sums the result onto the feature map. After a convolution layer, pooling layer is added to continuously reduce the dimensionality to reduce the number of parameters and computation in the network. This shortens the training time and controls overfitting. [6].

F. Long Short Term Memory (LSTM) Recurrent Neural Network:

Recurrent Neural Networks are a type of Artificial Neural Network with loops in the node allowing them to store the information. It can be thought of as multiple copies of the same network, each passing a message to a successor. RNN has issue of long term dependencies where the gradients propagated over many stages tend to either vanish or explode [7]. This is resolved with the help of a special kind of RNN: Long Short Term Memory (LSTM).

LSTM prevents the problem of backpropagation from vanishing or exploding. LSTM avoids long- term time dependency problems by controlling information flow using input, forget and output gates [8]. LSTM processes input sequences in forward and backward directions and is able to summarize temporal information from past and future contexts, so now both past and future dependency information are used to capture the temporal information [9]. Due to inherent sequential nature of sensor data, LSTM are well -suited for classification of text questions into ‘sincere’ or ‘insincere’ category.

G. K-Means Clustering

K-Means Clustering is an unsupervised classification algorithm which can group similar data points together and discover underlying patterns. It does this by looking for a fixed number (k) of clusters in a dataset.

Clusters can be defined as group of data points with some kind of similarities. K-Means algorithm works by assigning k number of imaginary centroids and each data point is assigned to these centroids by reducing the in-cluster sum of squares. The K-Means algorithm starts with random centroid assignment and then performs iterative (repetitive) calculations to optimize the positions of the centroids. It stops when either the centroids have stabilized or maximum number of iterations reached [20].

III. Related Work

Recent research on text classification tasks makes use of neural networks which has shown to outperform methods based on the bag-of-words model. Instead of text, they take input converted from text to numeric known as word vectors [10]. These word vectors can be learned either using skip-gram [11] or Glove [12] methods. These word vectors learn the semantic relationships between words such that in the embedding space. To learn document embeddings from these word vectors, Le and Mikolov [13] use distributed bag-of-words (DBOW) approach while Joulin et al. [14] computes the average of the word and subword vectors of a document to train a linear classifier. To extract sentence level features for text classification task, Kim [15] uses shallow CNN with max-pooling on top of pre-trained word vectors. They also observe that learning task-specific vectors through fine-tuning leads to better classification accuracy. Similarly, LSTM network pre-trained using language model parameters or sequence autoencoder parameters is used by Dai and Le [16] for various text classification tasks. Recently, it was shown by Johnson and Zhang [17] that a CNN with dynamic max pooling layer can effectively use the word order structure when trained using one-hot encoding representation of input words. They also learn multi-view region embeddings through semi-supervised learning and incorporate them as additional features to further improve the model's performance [17]. Similarly, they also perform semi-supervised experiments using a simplified LSTM model which also takes one-hot encoding of words as input [18] [19]. Also, since this is a Kaggle competition, we also looked at various kernels to gain idea about different approached to this problem as part of our background research.

IV. Project Description & Experiments

The project was carried out in various steps, such as Exploratory Data Analysis, Sentiment Analysis, Handling Data Imbalance, Fitting Supervised Machine learning Classification Models, Fitting unsupervised classification model and finally performance comparison.

A. Exploratory Data Analysis:

The aim of conducting Exploratory Data Analysis (EDA) was to gain an idea about the dataset characteristics. We wanted to get an idea about the proportion of the sincere vs insincere questions. The pi-chart in Figure 1 below shows that the dataset is quite skewed, the insincere texts comprise of only 6.2% of the entire dataset.

Apart from the data imbalance, we also visualized a word-cloud to get an idea about the most frequent words in the texts.

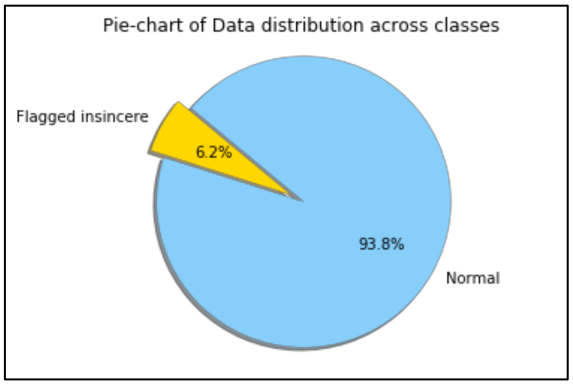


Fig 1: Pie-chart indicating data imbalance

As can be seen in the word-cloud in Figure 2, “India”, “Will” “One”, “think”, “mean” are some of the most highly used words in all the texts.

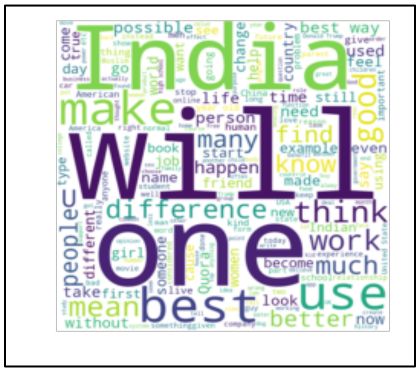


Fig 2: Word-cloud of Quora question text

Clearly, getting to know single most frequent words in the texts gives us an extremely vague idea about the kind of questions or the sentiments associated with them. The natural next step was to carry out sentiment analysis of the texts.

B. Sentiment Analysis

In the first step of the Sentiment Analysis, we decided to find the most commonly occurring words in the sincere and insincere question texts As we can see in Fig [3], the most commonly occurring words in the sincere texts are “like”, “use”, “people”, “good”. While otherwise, the most commonly occurring words in the insincere texts Fig [4] are “people”, “woman” and “trump”. Trying to achieve a better understanding of the contexts in which these words were used, we also found the top bigrams and trigrams in the sincere and insincere questions [Fig [5][6][7][8]]. The top bigrams in sincere texts were “year old”, “look like”, and “united states”, and for insincere texts they were “donald trump”, “white people”, “black people”. We could see that bigrams made more sense than just looking at single words, and it also made sense intuitively, especially in case of the insincere bigrams and also trigrams.

C. Data Preprocessing

a. Train-Test Split

Before fitting classification models, we split our dataset into two unequal parts, namely training and test sets. The training set was randomly chosen from the dataset and contains 80% of the original set, while the test contains the rest 20%.

b. Handling Imbalance

As observed in our exploratory analysis, the data was highly skewed, meaning, the number of insincere questions in the dataset are significantly more than that of sincere questions. Since model fitting does not do well on skewed data, we decided to handle the data imbalance issue. There are two possible ways of handling data imbalance, one is to draw a random sample form the majority class roughly the same size as minority class. Since the minority class in our dataset constitutes only 6.2% of the dataset, this undersampling technique would have greatly reduced the size of our dataset, and hence prove challenging when fitting classification models, especially models like Neural Networks which require large amount of data to perform well. Hence we decided to go with an oversampling technique, known as SMOTE (Synthetic Mean Oversampling Technique) [3]. After SMOTE oversampling on the training set, our dataset had roughly the same representation of both the sincere and insincere classes. Fig [9].

c. Vectorization

Since our aim was to fit mathematical models to text data, we needed an accurate quantitative representation of the text data. This process of converting text to numerical representation, that preserves its relative signification, and association with other words, is known as vectorization. For our purposes, we used TF-IDF (Term Frequency - Inverse Document Frequency) vectorization. It's basically a technique for representing a word as a numerical value that indicates its importance in the document.

Another way of converting and mapping vocabulary of texts to vectors of real numbers is through word-embeddings. For this paper, we are using glove.840B.300d which is a pre trained word vector with 840B tokens, 2.2M vocab, cased and 300d vectors. It was formed using the GloVe is an unsupervised learning algorithm for obtaining vector representations for words [21]. GloVe creates word vectors that capture meaning in vector space and it takes advantage of global count statistics instead of only local information. Unlike word2vec, which learns by streaming sentences, GloVe learns based on a co-occurrence matrix and trains word vectors so their differences predict co-occurrence ratios. It weighs the loss based on word frequency [22].

D. Model Fitting

a. Logistic Regression

Logistic Regression is one of the simplest classification models in terms of fitting and analysis. Hence we fit two Logistic Regression models on the training data, after TF-IDF vectorization. one before oversampling, and one after, since we wanted to find if oversampling indeed significantly affects the performance on the test set or not. We observed that although the training set before oversampling does not fit a high-performing model, somehow the oversampled training set fails to achieve even that performance.

Model type	Accuracy	Precision	Recall	F-1 score
Before SMOTE	0.94	0.70	0.22	0.33
After SMOTE	0.89	0.33	0.62	0.43

Table 1: Logistic Regression Classification Report

Logistic Regression has a lot of shortcomings, one of the biggest being it's highly sensitive to outliers. Hence we proceeded to fit more nuanced and complex models for our data. Also, since SMOTE failed to perform better than the original data, we decided to use the original training set for fitting further models.

b. Support Vector Machines

After Logistic Regression, we fit a Support Vector Machine (SVM) Classifier to our training data. We fit 3 SVM classifier models, with linear and polynomial kernels of degrees 2 and 3 respectively. Linear kernel gives the best results. The performance drops significantly when we try to fit SVM models with higher degree kernels, such a quadratic or cubic kernels, which signals overfitting.

Kernel type	Accuracy	Precision	Recall	F-1 score
Linear	0.948	0.67	0.25	0.37
Poly-2	0.935	0.00	0.00	0.00
Poly 3	0.940	0.00	0.00	0.00

Table 2: SVM Classification Report

c. Feed Forward Neural Networks

For the purpose of fitting Neural Networks, we used Multi Layer Perceptrons (MLP) of varying number of layers and different layer sizes. Since fitting of neural network models is a time consuming process, and also requires high computation capabilities which was beyond our scope for this project, we decided to work with a fraction of our dataset. For fitting MLP, we randomly drew 3% of the dataset and carried out train test split. We that vectorized the datasets using TF-IDF vectorization method. We fit the models and observed the various classification scores as illustrated in Table 3. We could see that the best result is obtained by a Neural Network of with 3 hidden layers, the layers having 10, 5, and 2 neurons respectively. We did not see any encouraging outcome of this model, hence we decided to explore further with Neural Networks of more complex natures, which are more suitable for text classification.

# of Hidden Layers	Accuracy	Precision	Recall	F-1 score
2	0.06	0.06	1	0.11
3	0.849	0.25	0.74	0.37
4	0.81	0.21	0.77	0.33
5	0.87	0.28	0.69	0.39
6	0.86	0.26	0.72	0.38

Table 3: MLP Classification Report

d. CNN

We fit a Convolutional Neural Network (CNN) model with 3 1-D convolutional layers, fitted on training data which were vectorized using Glove word-embeddings. Generally CNN models are used for image classification, and 2-D Convolutional layers are widely used. But since the vectorized data can be considered 1-dimensional, 1-D layers would suffice for our purpose.

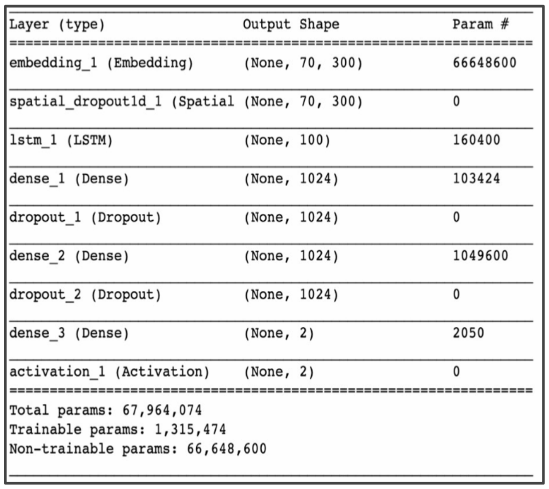
Threshold	Accuracy	Precision	Recall	F-1 score
0.1	0.91	0.41	0.88	0.56

0.2	0.94	0.55	0.74	0.63
0.3	0.95	0.63	0.61	0.62
0.4	0.95	0.67	0.47	0.55
0.5	0.95	0.70	0.38	0.49

Table 4: CNN Classification Report

e. LSTM-RNN

Since Long Short Term Memory Recurrent Neural Networks perform well with text data due to their sequential nature, we fit an LSTM RNN model with the specifications as described Table 5.



```
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 70, 300)           66648600
_________________________________________________________________
spatial_dropout1d_1 (Spatial (None, 70, 300)           0
_________________________________________________________________
lstm_1 (LSTM)                (None, 100)               160400
_________________________________________________________________
dense_1 (Dense)              (None, 1024)              103424
_________________________________________________________________
dropout_1 (Dropout)          (None, 1024)              0
_________________________________________________________________
dense_2 (Dense)              (None, 1024)              1049600
_________________________________________________________________
dropout_2 (Dropout)          (None, 1024)              0
_________________________________________________________________
dense_3 (Dense)              (None, 2)                 2050
_________________________________________________________________
activation_1 (Activation)    (None, 2)                 0
=================================================================
Total params: 67,964,074
Trainable params: 1,315,474
Non-trainable params: 66,648,600
_________________________________________________________________
```

Table 5: LSTM-RNN Model Summary

Similar to our CNN model, we tried various threshold values, and found that for a threshold of 0.4 we get the best F1 score (0.65). We also get a very high accuracy of 0.959 corresponding to this threshold.

f. Clustering

We also wanted to see if unsupervised classification works well in this case. So for the purpose of this, we performed K-Means Clustering on the question text, ignoring the labels, viewing it as an unsupervised machine learning problem. However, we did not achieve any encouraging response. The accuracy (0.89), precision (0.0126), and recall (0.009) values were very poor.

V. Conclusion

As expected, LSTM-RNN model with Glove embedding gives the best performance on our dataset, since it's best suited for data of sequential nature. Since, the data-set was massive, we needed high computation power and time to try different layers and tune complex neural networks to improve our score even further. We tried using cloud resources with GPU access like "Google Collaboratory" but since we were working with Glove word embeddings which in itself is a huge file, the notebook kept crashing. As LSTM takes care of long-term time dependencies, it performed well on this data-set as it tries to find the meaning of the whole sentence by keeping all the words in consideration. Further scope for this project includes trying various configurations of layers for LSTM RNN models, using CNN and LSTM together and working with different word embeddings to compare which embedding gives the best performance on this data-set.

VI. References

[1] Quora Homepage. URL: https://www.quora.com

[2] Quora Insincere Questions Classification. URL: https://www.kaggle.com/c/quora-insincere-questions-classification

[3] N. V. Chawla et. al. SMOTE: Synthetic Minority Over-sampling Technique, June 1, 2002, URL: https://doi.org/10.1613/jair.953

[4] C. M. Bishop, Pattern Recognition and Machine Learning, 1st Ed, 233 Spring Street, NY: Springer, 2006.

[5] A.I.Wiki. A Beginner's Guide to Convolutional Neural Networks (CNNs). URL: https://skymind.ai/wiki/convolutional-network

[6] Daphne Cornelisse. An intuitive guide to Convolutional Neural Networks. URL: https://medium.freecodecamp.org/an-intuitive-guide-to-convolutional-neural-networks-260c2de0a050

[7] Y.Bengio, P.Simard and P.Frasconi, "Learning long -term dependencies with gradient descent is difficult," IEEE transactions on neural networks, vol. 5, no. 2, pp. 157–166, 1994.

[8] S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural computation, vol. 9, no. 8, pp. 1735–1780, 1997

[9] Christopher Colah. Understanding LSTM Networks. URL: http://colah.github.io/posts/2015-08-Understanding-LSTMs

[10] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2, NIPS'13, pages 3111–3119, USA. Curran Associates Inc.

[11] Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32, ICML'14, pages II–1188–II–1196. JMLR.org.]

[12] Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1532–1543, Doha, Qatar, October. Association for Computational Linguistics.

[13] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2, NIPS'13, pages 3111–3119, USA. Curran Associates Inc.

[14] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of tricks for efficient text classification. In Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers, pages 427–431, Valencia, Spain, April. Association for Computational Linguistics.

[15] Yoon Kim. 2014. Convolutional neural networks for sentence classification. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1746–1751, Doha, Qatar, October. Association for Computational Linguistics.

[16] Andrew M. Dai and Quoc V. Le. 2015. Semi-supervised sequence learning. In Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2,

NIPS'15, pages 3079–3087, Cambridge, MA, USA. MIT Press.

[17] Rie Johnson and Tong Zhang. 2015a. Effective use of word order for text categorization with convolutional neural networks. In Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 103–112, Denver, Colorado, May–June. Association for Computational Linguistics.

[18] Rie Johnson and Tong Zhang. 2016. Supervised and semi-supervised text categorization using lstm for region embeddings. In Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48, ICML'16, pages 526–534. JMLR.org.]

[19] dsachan, manzilz, rsalakhu. Investigating the Working of Text Classifiers. https://aclweb.org/anthology/C18-1180.

[20] Dr. Michael J. Garbade. Understanding K-means Clustering in Machine Learning. URL: https://towardsdatascience.com/understanding-k-means-clustering-in-machine-learning-6a6e67336aa1

[21] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014

[22] keitakurita. Paper Dissected: "Glove: Global Vectors for Word Representation" Explained. URL: http://mlexplained.com/2018/04/29/paper-dissected-glove-global-vectors-for-word-representation-explained

VII. Appendix

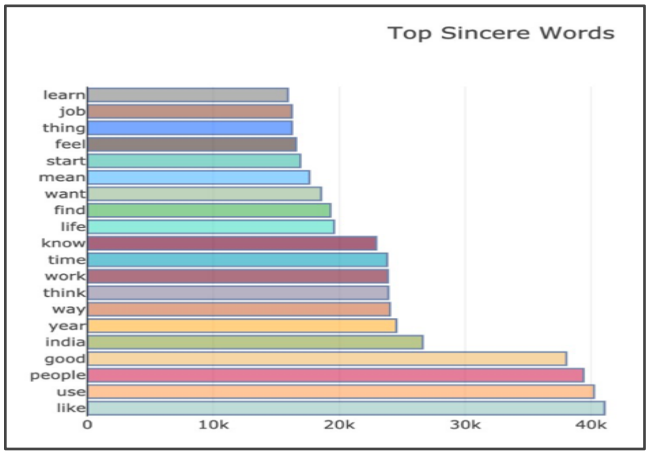


Fig. 3. Top Sincere Words

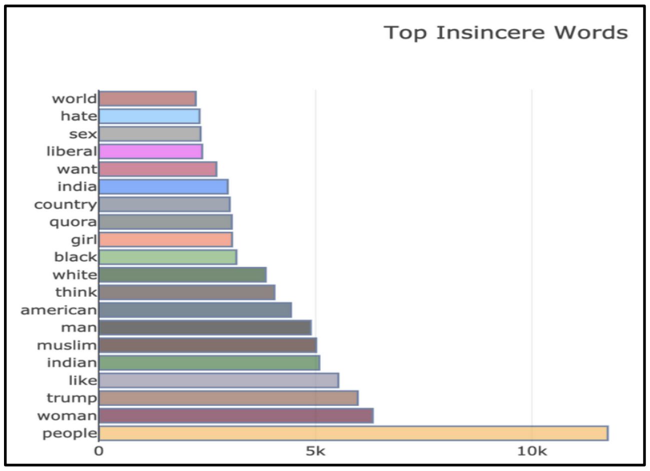


Fig. 4. Top Insincere Words

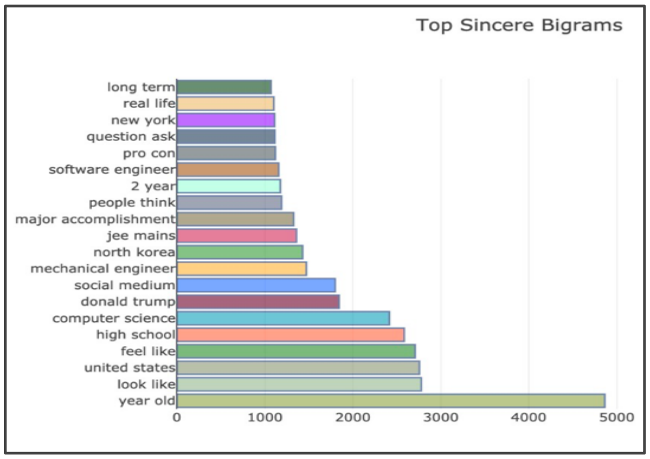


Fig. 5. Top Sincere Bigrams

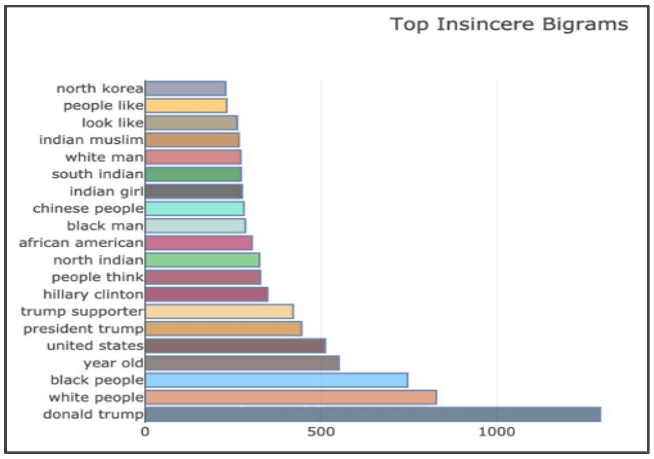


Fig. 6. Top Insincere Bigrams

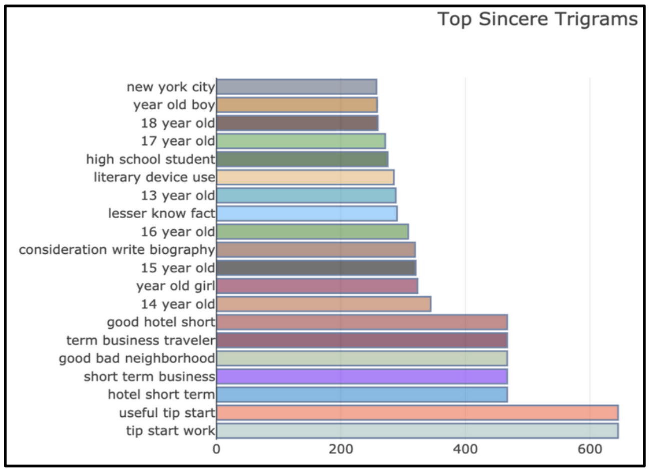


Fig. 7. Top Sincere Trigrams

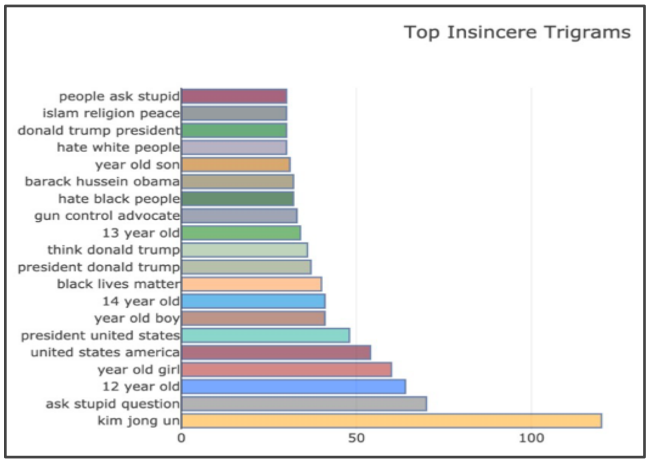


Fig. 8. Top Insincere Trigrams

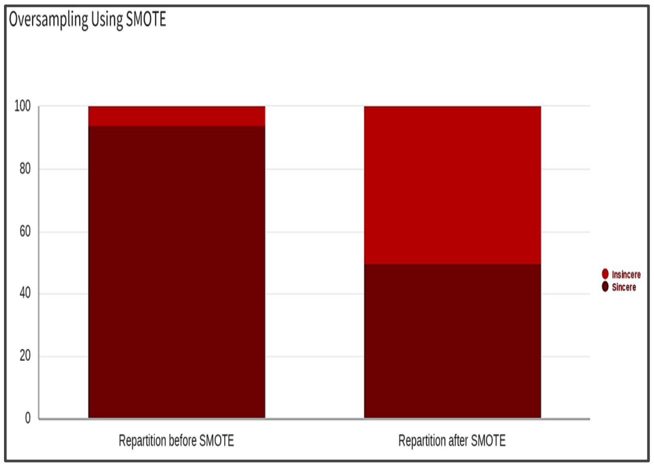


Fig. 9. Oversampling of minority class using SMOTE