A Project Report for CSN-300 (Lab Based Project) of Spring Semester 2020-2021

On

Insincere Question Classification

Submitted by

Diksha (18114019)

diksha@cs.iitr.ac.in

Kavya Barnwal (18114039)

kbarnwal@cs.iitr.ac.in

Karan Singh (18114035)

ksingh1@cs.iitr.ac.in

Supervised by

Prof. R. Balasubramanian

Department of Computer Science and Engineering

Indian Institute of Technology (IIT) Roorkee

May 4, 2021

Abstract

With the increasing growth of social media platforms and improving the scope of connectivity among people worldwide, there is also a rising spread of misinformation and toxicity. And it is posing a great challenge for these platforms to handle toxic and divisive content. Quora, being one of such platforms, encourages people to ask questions, even anonymously, and connect with people who can collaborate with their unique insights and helpful answers. While most of the questions asked with an intention to look for their concerned answers and are sincere, there are few questions that can be said insincere questions - which are founded upon false premises to confuse general audience, or asked with an intention to spread propaganda or make statement rather than look for helpful answers.

In this project, we present the solutions for filtering such toxic or insincere questions for better connected social communities. We tried various Natural Language Processing techniques for filtering insincere questions asked on Quora, taking the dataset from Kaggle [1] Challenge [2] hosted 2 years ago. The methodology can be adopted by other social media platforms as well for the detection of toxic and divisive content that may be posted mistakenly or with malicious intentions.

We described our Problem Statement in Sec. 3, and defined the task as text-classification of questions into sincere and insincere, and tried to put a short review of the literature present around Natural Language Processing Text Classification Tasks in Sec. 2, and then presented our derived insights from the dataset in Sec. 4. All of our methods that we tried in order to get best performance in this classifications task is mentioned in Sec. 5, and then presented the results obtained from our experiments in Sec. 6, and finally put our conclusions with the ideas that we can try in future if we were to continue the project in Sec. 7.

Contents

1 Introduction

In this modern era, most of the work is done online. There are many online platforms for people to ask questions or discuss topics or get answers to what they are looking for. While, for the most part, these questions are asked in good faith, some people post questions that are insincere or problematic. To overcome this drawback , the task is to build an automatic classifier which takes a user generated question as an input and classifies that question as sincere or insincere. The definition of what qualifies as an insincere question may vary according to people , but here are some parameters to identify one :

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

For this, we have made use of several Natural Language Processing techniques with Deep Neural Networks, CNN, GRU, LSTM, and BERT, a state-of-the-art NLP model for a wide variety of NLP tasks.

2 Literature Review

An end-to-end text classification model consists of following building blocks :

1. **Training input:** It is the input examples (which in our case is the user generated input text) which gives the supervised learning model the capability to learn and classify the input accurately.
2. **Feature Vector:** This is the vector that contains information describing the characteristics of the input data.
3. **Labels:** These are the categories in which we want to classify the inputs.
4. **ML/DL Algo:** It is the algorithm used to accomplish the task of classification .
5. **Predictive Model:** A model which is trained on the historical dataset which can perform label predictions.

The simplest model we have used for the classification is deep neural networks i.e., the vanilla neural network, with one neuron. A deep neural network basically consists of a number of hidden layers, each of which contains several neurons as shown in Fig. 1. The input is passed through these hidden layers, with each layer having its activation function, and we get the output at the end (which in our case is the class of whether the sentence is insincere or not).

CNN(Convolutional Neural Networks) are neural networks using feed forward method , where we use a filter/kernel of odd size and parse it across the input, in a patch by patch

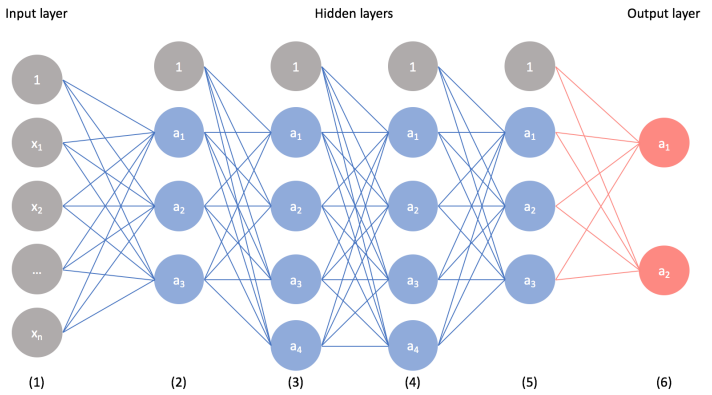


Figure 1: Example of Deep Neural Network with 4 hidden layers

manner and identify patterns in the data. Thus, it can extract features very efficiently from the patterns present in the input data in place of multilayer perceptrons due to its minimal preprocessing requirements and the fewer parameters required to train. By varying the kernels across the input and concatenating their outputs, we can detect patterns in the input. And as patterns can also be found in sentences. Therefore, CNNs can identify them in the sentence regardless of their position, which will help in identifying any toxic patterns/ expressions in the context of our classification. Also, training and inference time for CNN is faster due to possible parallelization if we compare it with recurrent neural networks like RNN, GRU, or LSTM.

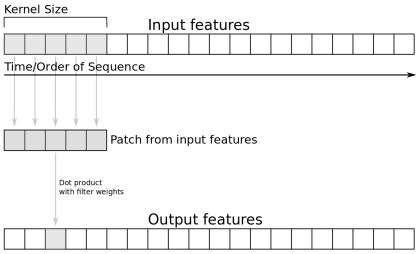


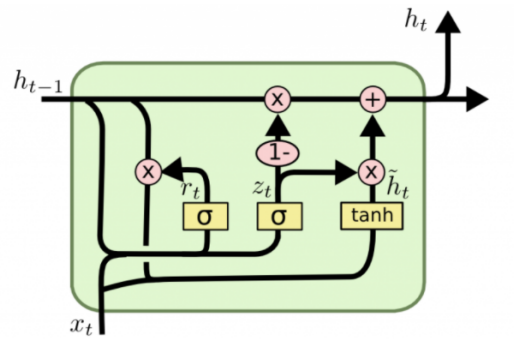
Figure 2: 1-dimensional convolution with a kernel of size 5

Recurrent neural networks(RNN) are a class of neural networks where a sequence of neural network blocks ,linked with each other are considered . Each block passes some information to the succeeding block. RNNs uses their internal state (memory) to process sequences of inputs. They are helpful in tasks related to sequential data, data where chronological ordering matters. Thus, due to its remarkable capability of memorizing sequential information, it is of great application in NLP tasks. In our case, the sequential data will be string of words as sequential data, and each recurring block will integrate new information (lexical and semantic) about the words, information that has been trained and distilled on a very large corpus of data.

But standard RNNs do not learn efficiently if there is a greater time lag between relevant input events and target signals. It greatly suffers in keeping the information for longer distances due to the vanishing gradient problem. LSTM, “Long Short-Term Memory” [3], is not affected by this problem as it can learn to bridge minimal time lags by enforcing constant error flow during backpropagation. LSTM uses recurring network with each recurring unit having one input layer, one hidden layer, and one output layer. Hidden layer contains memory cells and corresponding various gates, in which they uses operation like sigmoid or tanh activation or

Notation

h_t : hidden layer vectors.
x_t : input vector.
b_z , b_r , b_h : bias vector.
W_z , W_r , W_h : parameter matrices.
σ , $\tanh$: activation functions.



Feed-Forward

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t] + b_z)$$

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t] + b_r)$$

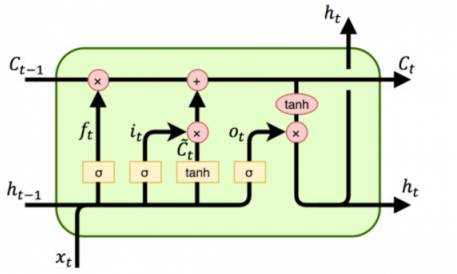
$$\tilde{h}_t = \tanh(W_h \cdot [r_t \odot h_{t-1}, x_t] + b_h)$$

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t$$

Figure 3: Recurring Unit of GRU

Notation

h_t , C_t : hidden layer vectors.
x_t : input vector.
b_f , b_i , b_c , b_o : bias vector.
W_f , W_i , W_c , W_o : parameter matrices.
σ , $\tanh$: activation functions.



Feed-Forward

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

$$\tilde{C}_t = \tanh(W_c \cdot [h_{t-1}, x_t] + b_c)$$

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t$$

$$h_t = o_t \odot \tanh(C_t)$$

Figure 4: Recurring Unit of LSTM

multiplication or addition. Each memory cell's internal architecture guarantees a constant error

flow, which helps in overcoming the vanishing gradient problem, thus bridging the long-time lags.

There is also a simple version of LSTM which nearly gives the same results in several NLP tasks, namely "Gated Recurrent Unit"(GRU) [4], which is mostly similar to LSTM, except that it has just two gates reset and update gates while LSTMs has three gates, input, output, and forget gates. Feedforward equations and notations of the recurring units of GRU can be understood from the Fig. 3, and that of LSTM it is explained in Fig. 4

BERT (Bidirectional Encoder Representations from Transformers), originally published by researchers at Google AI Language [5], is state of the art model for several NLP tasks. BERT provides context-aware dense vector representations by making use of a deep pre-trained neural network with the Transformer architecture, pretrained with two tasks Masked Language Modeling (MLM) and Next Sentence Prediction (NSP). Its architecture is basically an Encoder stack of transformer architecture. And, a transformer architecture [6] is an encoder-decoder network that uses self-attention on the encoder side and attention mechanism on the decoder side.

3 Problem Statement

Given our dependency on social media and online platforms in this age , these platforms play their part in both helping our problems but also spreading hatred and toxicity too. And, thus there is a need of automatic classifier which can detect insincere or toxic questions. For that we define our problem as a text classification task in which we have to classify a user generated question as insincere or not.

Given the input $\mathbf{x}$ (which will be a text, a sentence or question, that people posts on the online platforms), find a model $\mathbf{f}$ which will output $\mathbf{y}$, a label denoting whether the question is insincere(1) or sincere(0).

4 Data Analysis

Our Dataset is taken from Kaggle [1] and it consists about 1.3 millions examples. It consists of questions text asked on Quora and with a label of 1(classified as insincere questions) or 0(classified as sincere questions).
Each question text is string of words and our dataset has following characteristics:
Average word length of questions is 13.
Max word length of questions is 134.
Min word length of questions is 1.

Average number of unique words in questions is 12.
Average number of punctuation in questions is 2.
Average character length of questions in train is 71.

It is also quite interesting to compare the average word length and average unique word counts in sincere and insincere questions.
Average word length of sincere questions is 13. While,
Average word length of insincere questions is 17.

And, Average number of unique words in sincere questions is 12. while
Average number of unique words in insincere questions is 16.

So, it can be seen that the insincere questions are more longer with more unique words in comparison to sincere questions.

Also, this Dataset is a classic example of skewed dataset as it contains mostly sincere questions, and very few insincere ones, about 6.19% only, as in Fig. 5, because most of the questions asked on social platforms are genuine.

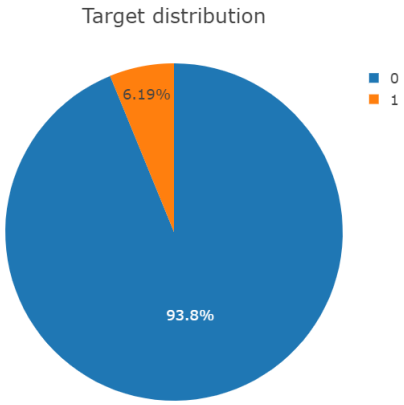


Figure 5: Distribution of Insincere and Sincere questions

There is a large number of unique words in this dataset,as many as 284445 in its raw form, so it would be intriguing to look at the most frequent words used in sincere and insincere questions, which can be seen in the below diagram Fig.6 , with their word counts.

It can be clearly observed that there are few common words like 'people', 'will', 'think', etc in both types. And excluding such words, 'best', 'good', 'someone', 'use', etc are the most frequent for sincere questions, and words like 'trump', 'women', 'white', 'muslim', 'black', etc are the most frequent for the insincere questions, clearly trying to make some political, social statements or pointing towards some community.

5 Methodology

In this part, we will try to explain all the experiments that we were able to run to get better results for the above-given problem statement in Sec. 3. For all of our Neural Networks, we used tensorflow [7] and keras [8] libraries for Tokenization of the words and generating Embedding vectors for the words in the questions of the dataset. As we had mentioned, the average word length in questions of our dataset is around 13. Still, the maximum word length is as long as 134, so we decided to go with a maximum word length of 32 for each question by either truncating the words for more extended questions or padding zeros for shorter questions.

One more thing worth mentioning here is that due to the skewed nature of the dataset, we find much better results by lowering the threshold value in predictions after the final sigmoid layer from 0.5. It improves the f1-scores by a fair margin for almost every model we used.

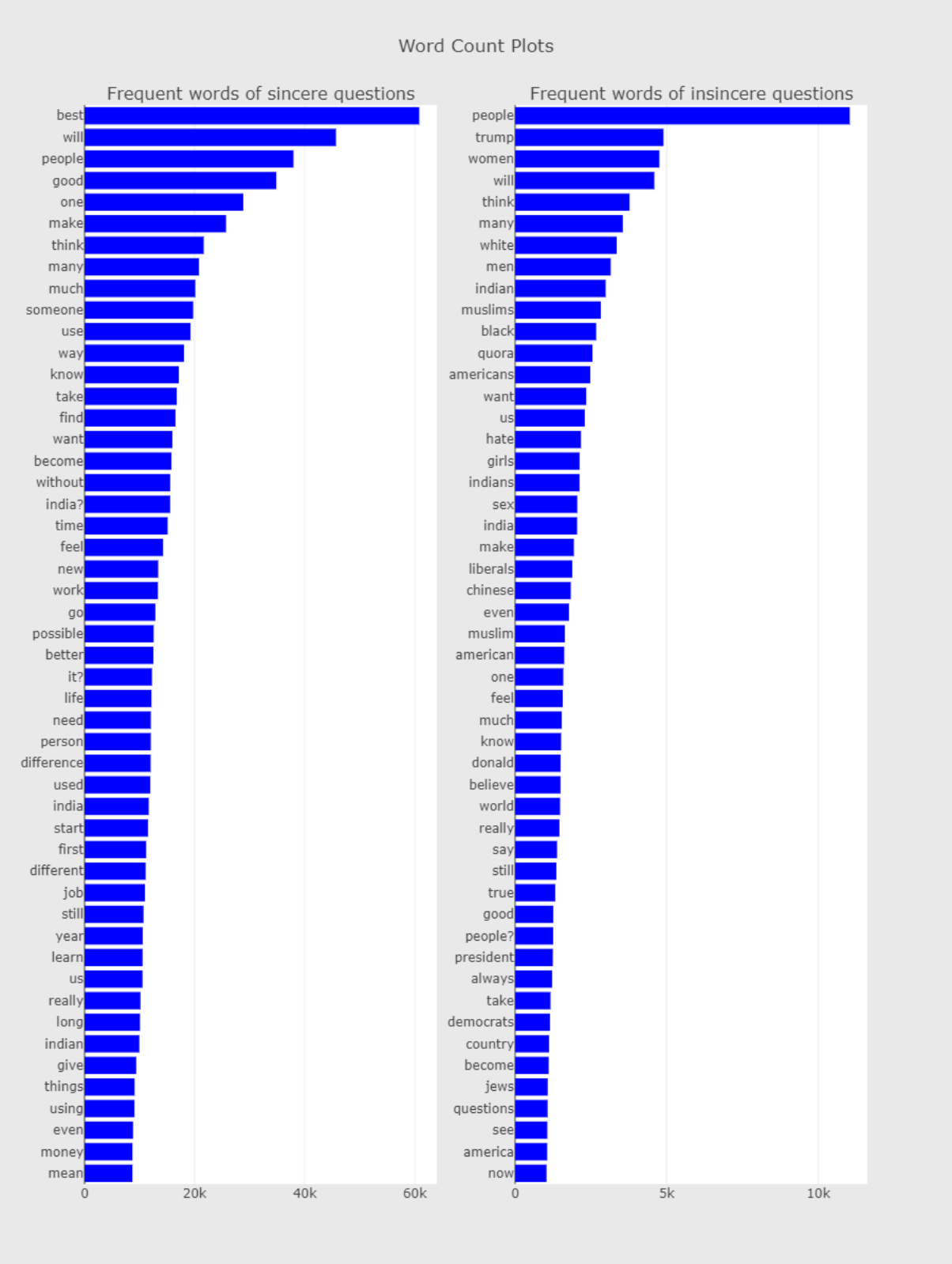


Figure 6: Most frequent word in Sincere and Insincere questions

Our complete code-base can be found on github. We would like to mention that although we tried a lot of experiments to improve our f1-score, which we'll be presenting in the next section of Simulation results6, but we were limited because of the COVID-19 pandemic, and

our compute budget was also minimal. And, we did all of our experiments with the help of google collaboratory only. Majorly we experimented with six models that were mentioned below, and a comparison of the results is shown in Table 1.

5.1 Baseline

We already mentioned in Data-analysis in Sec. 4 that the used Kaggle dataset is a skewed dataset, and it has very few insincere labeled examples, about 6.19% only.

So, we used our baseline model as simple $\mathbf{f(x)=0}$, i.e., give output 0 for every given input x, which still brings us the accuracy of 93.8%. Although the accuracy for this extremely simple baseline seems pretty good, it barely classifies as a good model, as it's predicting insincere questions 100% wrongly, and its recall value would be 0.

5.2 Logistic Regression

In this method, we used a neural network in which we first used the embedding layer generating word embedding of 100-dimensions, then average-pooled its output, and then simply classified the pooled output with just a single neuron.
Although the model was very simple to train, it also took fair training time due to this 1.3M large dataset. We split the dataset into 80:20 for training and test purpose and then did a validation split of 20%. We used the binary entropy loss function and Adam optimizer for this and all rest of the models for a few epochs.

5.3 Multi-layer Perceptron

Trying to improve on the previous single neuron classifier 5.2, we used multilayer perceptron by introducing one hidden layer, with 32 neurons, in the neural network after the Average pooling layer, with 'relu' [9] activation function. The model did improve our f1-score on the test set by a fair amount of change. Multi-Layer Perceptrons can be picturized as shown in Fig. 1

5.4 CNN : Convolutional Neural Network

Trying to explore other neural nets, we also tried 1D Convolutional Neural Networks. It is applied just after the embedding layer with a kernel of size 5, and then flattened its output for further hidden layer of 512 units, then another layer of 10 units, with relu [9] activations for each, after which we finally passed it to the final sigmoid layer for binary classification. 1D Convolutions can be understood from the Fig. 2.

5.5 GRU : Gated Recurrent Unit

Due to the great applications of RNNs in NLP tasks, we built a neural network with Gated Recurrent Unit [4] to improve the results. We used Bidirectional neural architecture with GRU, i.e., processed the input in both forward and backward directions and then concatenated the outputs of forward and backward parsing. Then passed its output to a hidden layer of 32 units, with relu [9] as activation function, then used the final sigmoid layer. The recurring unit of GRU can be understood from the Fig. 3, with its feedforward equations and notations.

5.6 LSTM : Long Short Term Memory

For further improvements, we also tried Long Short Term Memory [3]. We knew that GRUs are more efficient than LSTMs, due to their simple structure and less number of gates(GRU has two gates, reset and update gates while LSTM has three, input, output, and forget gates). Thus, LSTM can capture more information than GRU and can remember longer sequences due to its more complex architecture. The complex structure of the recurring unit LSTM is explained along with its notations, and feedforward gate equations are explained in Fig. 4. Also, LSTM is illustrated in great detail in Colah's blog [10]. Here, we used the same neural network stack as in GRU 5.5 model, only replacing GRU with LSTM.

5.7 BERT : Bidirectional Encoder Representations from Transformers

Lastly, we thought it'd be good to experiment with pre-trained embeddings for better performance. There are several pre-trained word embeddings like word2vec, GloVe, FasText, which don't utilize contextual information. At the same time, there are also pre-trained word models like ELMo and BERT that can generate word embeddings for a word that captures the context of a word - that is its position in a sentence. But due to limitations of computing budget and time we could only experiment with BERT. BERT (Bidirectional Encoder Representations from Transformers) provides dense vector representations for natural language by using a deep, pre-trained neural network with the Transformer architecture. A transformer architecture [6] is an encoder-decoder network that uses self-attention on the encoder side and attention on the decoder side.

Thus, we used the pre-trained BERT base model from TF-Hub, and finetuned it with our dataset for our task. After the embeddings generated by the BERT layer, we just used the pooled output and passed it to a single neuron for classification, using the dropout regularization technique with the rate of 0.20.
Due to the minimal computing power that we have, we weren't able to train this with the complete 1.3M dataset, and we had to use only 0.10% of our dataset and finetuned the network for one epoch but still, we got our best f1-score of 0.659.

6 Simulation Results

In this part, we'll present the results obtained in our experiments.
We run each model only for few epochs, ranging from as low as 2 to as high as 20. It was observed that while the training accuracy improved after 20, but the validation accuracy started deviating and decreasing after few epochs, clearly indicating overfitting of our model to the training dataset. The f1-score after 20 epochs comes very low compared to what we received after 4 epochs only. It can be seen from the accuracy plot of the CNN model trained for 20 epochs, in Fig.7

One thing we experimented with is lowering the threshold value. As our dataset is highly skewed, there's a huge chance of neural networks learning to predict 0 for every input, leading to poor probability scores for insincere questions(which should be labeled as 1). Hence, we experimented if lowering the threshold from 0.5 would help or not, and as per our expectations, when we lowered the threshold value, results changed drastically in almost all cases.

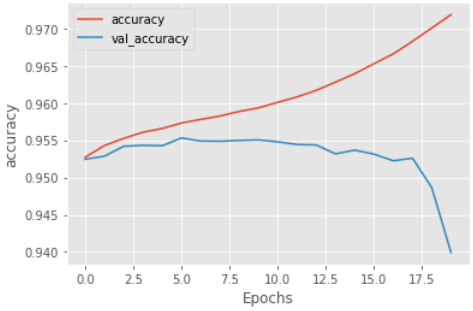


Figure 7: Accuracy plot for 20 epochs

For example, let us have a look at the confusion matrix for the same predicted output, for two different thresholds, Fig. 8 shows confusion matrix when the predicted probabilities classified into 0/1 with a threshold as 0.5, and Fig. 9 shows confusion matrix when the predicted probabilities classified into 0/1 with a threshold as 0.26. F1-scores obtained for them are 0.5394 and 0.6057, respectively.

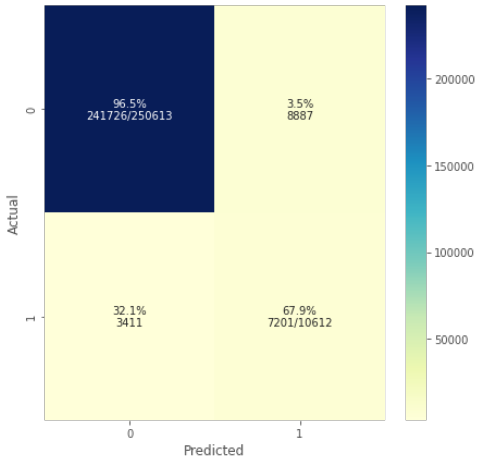


Figure 8: Confusion Matrix for Threshold=0.50

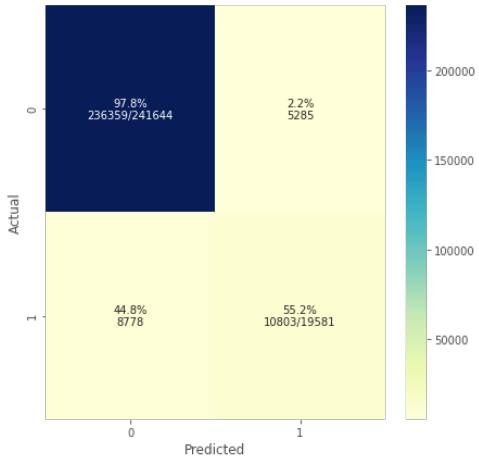


Figure 9: Confusion Matrix for Threshold=0.26

Models	Accuracy	Precision	Recall	F1-Score
Baseline	0.938	-	-	-
Logistic Regression(threshold=0.50)	0.953	0.68	0.45	0.539
Logistic Regression(threshold=0.26)	0.946	0.55	0.67	0.606
Multi-layer Perceptron(threshold=0.50)	0.955	0.70	0.48	0.569
Multi-layer Perceptron(threshold=0.23)	0.947	0.56	0.70	0.624
Convolutional Neural Network(threshold=0.50)	0.950	0.60	0.61	0.606
Convolutional Neural Network(threshold=0.40)	0.946	0.55	0.70	0.615
Gated Recurrent Unit(threshold=0.50)	0.955	0.67	0.55	0.603
Gated Recurrent Unit(threshold=0.28)	0.950	0.58	0.69	0.629
Long Short Term Memory(threshold=0.50)	0.956	0.67	0.55	0.605
Long Short Term Memory(threshold=0.28)	0.951	0.59	0.67	0.629
Pretrained BERT(threshold=0.50)	0.956	0.64	0.67	0.655
Pretrained BERT(threshold=0.44)	0.955	0.62	0.70	0.659

Table 1: Comparison Table for different neural networks we experimented with

Results of all the neural networks, which are mentioned in Sec. 5, can be seen from Comparison Table 1. Our main metrics for judgment of models is f1-score as that of the original competition [2]. We also decided to put the scores with and without moving the threshold from the standard 0.50 value. And, as per our theory of models learning to predict 0 every time, it can be clearly observed that we have to lower the threshold value in each case for the optimal f1-score for almost all models. Hence, it can be said that a skewed dataset hurts the confidence score of neural networks.

7 Conclusions and Future Work

Our major objective for this project was to classify the questions that were asked on Quora and label them as sincere or insincere. This methodology can also be adopted by other social media platforms to detect problematic and insincere posts and filter them out for a better socially connected world. For that, we run a variety of classification models and did several optimizations for them. And, the overall results that we got can be said reasonably successful. Almost all of our models, after doing several optimizations, performed nearly the same, as shown in the comparison table1. Still, the models which performed best other than pre-trained BERT were LSTM Sec.5.6 and GRU Sec.5.5 with nearly the same F1-scores of 0.62922 and 0.62920, followed by MLP Sec. 5.3 with F1-score of 0.62364 and CNN Sec. 5.4 with F1-score of 0.61489. Also found from our experiments that modifying the decision boundary or threshold helps to improve the F1 score for skewed datasets. Also, BERT-base, with a single neuron fine-tuned on 130k examples, gave the best f1-score of 0.659.

If we got the opportunity and enough resources to continue this project in the future, we will definitely try a variety of things for improvements, as this time we were limited by time and resources and COVID19 pandemic also put us in a bad situation. Although we have already tried a lot on preprocessing of this text dataset, such as converting abbreviations in text, removing URLs, HTML tags, punctuations, unnecessary spaces and escape characters, and did decontraction for words like I'm to I am, which also played a major role in improving results, but we wanted to see what other methods we can try for further improvements. We

would also like to try using other embedding schemes; moreover, we will try to look for other pre-trained embeddings, such as Word2Vec, GloVe, or FastText, as they had proved to be very useful in NLP tasks.

Finally, given more time and more computing resources (e.g., additional powerful GPUs), we would love to see what results BERT can deliver, if it is trained on the complete size of data and trained for more epochs, as it has shown best f1-score despite training for just one epoch on 0.10% of the full dataset. We would also like to see how the model improves its performance after using multiple layers of CNNs, GRUs, and LSTMs and their combinations in a single neural net.

References

[1] "Kaggle:quora insincere questions classification," 2018.

[2] "Quora insincere questions classification," 2018.

[3] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, p. 1735–1780, Nov. 1997.

[4] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," 2014.

[5] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2019.

[6] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2017.

[7] "Tensorflow guide."

[8] "Keras guide."

[9] A. F. Agarap, "Deep learning using rectified linear units (relu)," *CoRR*, vol. abs/1803.08375, 2018.

[10] C. Olah, "Understanding lstm networks."