Identifying insincere questions on Quora: A comparitive study of contrasting architectures

Aashika Shetty*, Kedar Deshpande†, Shruti Sharan‡ and Vidhu Malik§
Department of Computer Science,
University of California,
Los Angeles
Email: *aashikavshetty@ucla.edu, †kedardeshpande@ucla.edu, ‡shruti5596@ucla.edu, §vidhu26@ucla.edu

Abstract—**This project proposes a solution to the problem of automated recognition of insincere questions on Quora (an American question-and-answer website where questions are asked, answered, and edited by Internet users in the form of opinions) through the development and evaluation followed by the subsequent comparison among machine learning models like Bidirectional Long Short-Term Memory (BiLSTM) Model, Bidirectional Gated Recurrent Unit (BiGRU) Model and 1-Dimensional Convolutional Neural Network. The problem statement and dataset are part of a recently conducted Kaggle competition [1]. The main focus is to build an end-to-end machine learning pipeline starting from preprocessing the dataset in a suitable format, representing the dataset in a form that can be mathematically assimilated by using several pretrained embeddings like GoogleNews and finally training the different models and evaluating them using different metrics. One of the major challenges which was not part of the problem statement that was dealt with while working on this problem involved scaling the solution to make it work for the gargantuan real-world dataset ($\sim 1,3M$ records in the training dataset) that was provided.**

Index Terms—**Toxic Content Detection, Text classification, BiLSTM, BiGRU, CNN, Deep learning, NLP, NLU**

I. Introduction

Quora is a website that encourages people to learn from each other. On this platform, people have the ability to ask questions as well as answer them while simultaneously connecting with contributors whose insights they appreciate. Quora's mission is to share and grow the world's knowledge base. Yet, users are sometimes exposed to content of questionable intent. The problem of getting safe content and handling the toxic and divisive content is of utmost importance. One of the key challenges in this effort is filtering out insincere questions.

A. Motivation

Quora aims to create a safe and productive environment for its users. In such an atmosphere, the users will be free to see answers to the questions that matter to them and share their knowledge in a meaningful and respectful manner.

An insincere question intends to make a statement rather than look for helpful answers. Such a question might be raunchy, biased or downright derogatory. These questions tend to make the readers uncomfortable. The questions that target specific groups make the individuals who identify as a part of the said group feel unwelcome in the community. Quora, being a platform that caters to different communities and age groups has set up its core principle ("Be Nice, Be Respectful") that requires that people treat other people on the site with civility, respect, and consideration.

Weeding out insincere questions is an important step in this direction. The nature of this problem makes its solution non-obvious. The detection of insincere questions is a difficult problem to solve because what classifies as an insincere question is highly subjective. Additionally, these questions bear striking resemblance to the sincere questions posted on the website. Moreover, the size of the data, as well as the expansive number of questions posted on the website, makes it infeasible to manually classify every question as 'sincere' or ' insincere'.

Quora is trying to solve this problem by jointly using both machine learning as well as manual content moderation. Along with this, they have come up with a crowdsourced approach to address this issue. Like the Netflix challenge, Quora turned to Kaggle to design a competition where the contributors developed their own models for prediction.

B. Problem Definition

The objective of this project is to develop a system that can identify insincere questions present on Quora by developing models to detect toxic and misleading content. The Quora Insincere Questions Classification competition is a natural language processing task. The idea is to design a predictive model using the training dataset provided by Kaggle and then generate predictions for an unlabeled test set which would be evaluated on metrics such as accuracy, binary cross-entropy loss, F-1 score, etc.

In order to allow for a more formal description of the algorithms, some terms and variables that will be frequently used in the following are defined: Given a collection of questions $Q = \{q_1, q_2, ..., q_n\}$, let $V = \{w_1, w_2, ..., w_n\}$ be the set of distinct words/terms in the collection. Then V is called the vocabulary.

The problem of classification, in this case, is defined as follows:

Assume a training set of questions $Q = \{q_1, q_2, ..., q_n\}$, such that each question q_i is labeled with a label l_i from the set $L = \{0, 1\}$ where 0 indicates a sincere question and 1 indicates an insincere question. The task is to find a classification model (classifier) f where

$$f\colon Q \rightarrow L \text{ such that } f\left(q\right) = l$$

can assign the correct class label to a new question q (test instance).

II. Related Work

Flagging of insincere questions on internet forums is an ongoing challenge for online communities. Significant work has been done in this domain in previous years. Some important works have been discussed below. [2] tries to answer the question- Is this Quora question sincere by making use of three variations of the BERT model: vanilla BERT, BERT + CNN, and BERT + Linear. Their implementation on BERT is based on the implementation by Hugging Face AI, which has available a pre-trained BERT model with a linear layer after the pooled output. It was identified that since BERT relies heavily on 'giveaway' words (e.g. profanity) in flagging questions, it occasionally misses questions that use more advanced vocabulary and that are phrased as smoothly as sincere questions.

Another compelling work in this domain was conducted by [3]. They added attention to their neural network and proposed an Attention-based Long Short-Term Memory Network present aspect-level sentiment classification . They concluded that this attention mechanism can concentrate on different parts of a sentence when different aspects are taken as input.

[3] show a baseline model for the same problem. The features are extracted from the text using TfidfVectorizer to convert the text into a tf-idf matrix, which lays a foundation for subsequent text similarity calculation, searching and sorting of the text, and other applications. They use CountVectorizer() for cleaning and get a word-frequency matrix. They use three models, logistic regression, Multinomial Naive Bayes network, and Bernoulli Naive Bayes. Though these models set a good baseline, it is shown that a deep learning approach is better suited for such problems as compared to classical methods.

Prior literature shows the impact of using Convolution Neural networks for text classification. In [4], the authors show the efficacy of using 1D Text Convolutions for classification. The paper discusses the various convolution related constructs that are useful in text classification: MaxPooling, Usage of RELU Activation, Association of filters with classes. Similar ideas have also been introduced in [5], [6].

[7], [8] discuss the implementation of text classification models with recurrent networks like BiLSTM and BiGRU. Uniform layered architecture for recurrent networks has been shown to be simplistic and useful. These works also provide an intuitive explanation for the usage of LSTM's in text classification and support their theory with well-structured results.

III. Background: Predictive Modelling

A. Recurrent Neural Networks

In the domain of Natural language processing, neural networks have found large acceptance due to the mere sizes of these models and their ability to learn inherent underlying

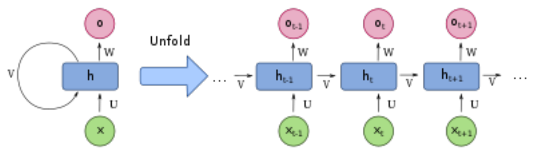


Fig. 1. Recurrent Neural Network

information that is difficult for classical algorithms to outperform. Recurrent Neural Networks are a type of neural networks that are used to process sequential data. [9] This is very useful in text classification problems since the sequence and order of words in a document, gives a lot of information about its content. For example if the words 'bark' and 'loudly' appear next to each other, it is likely that 'bark' refers to the sound made by a dog. Whereas if the words 'bark' and 'leaves' appear near each other, it is likely, that a the bark of the tree is being talked about. [3]

Basic RNNs are a network of neuron-like nodes organized into successive "layers", each node in a given layer is connected with a directed (one-way) connection to every other node in the next successive layer. Each node (neuron) has a time-varying real-valued activation. Each connection (synapse) has a modifiable real-valued weight. Nodes are either input nodes (receiving data from outside the network), output nodes (yielding results), or hidden nodes (that modify the data en route from input to output).

Recurrent neural networks can be thought of as multiple copies of the same network, that pass information along to each other. At each time step, there is some input to the model. After calculations have been made, the output gets sent forward to the next time step and is combined with other input. This is demonstrated in Figure 1. The output of each hidden state of the RNN is dependent upon the output from the previous hidden state, as well as the input to the current state. Being able to pass information along into different time steps is what gives RNN's the ability to make sense of sequential information.

One problem that arises in RNN's is whats known as the vanishing gradient problem. In neural network models, gradient descent is used to find the global minimum of a cost function that is optimal for the network. Information gets passed through the network from neuron to neuron and the error of the network is calculated, which is then back-propagated through the network to update weights. During propagation, every neuron that contributed to the error of the network must have its weighs updated, and since RNN's are sequential, we have to back propagate through lots of hidden layers of neurons [14]. When you multiply small weights over and over again, the value decreases very quickly, and this is the vanishing gradient problem. The problem with this is when you have a sequence of words, you can lose information about words that are far apart, and only track patterns of words that appear close to each other.

One way to solve this problem is to introduce ‘gating’ which is what Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRU) do. Gating simply helps the network know when to forget the input, and when to remember it for future time steps [4]. This solves the problem of the vanishing gradient. For our project, these two implementations of gated recurrent networks were experimented upon.

B. Convolution Neural Networks

Convolution Neural Networks is a class of deep, feed-forward artificial neural network and use a variation of multi-layer perceptrons designed to require minimal preprocessing. [10] CNNs were traditionally developed for computer vision, however they’ve recently been applied to various NLP tasks and the results have been promising.

The result of each convolution fires when a special pattern is detected. By varying the size of the kernels and concatenating their outputs, one detects patterns of multiple sizes (2, 3, or 5 adjacent words).Patterns could be expressions and therefore CNNs can identify them in the sentence regardless of their position.

Another important construct in convolution networks is that of pooling. Pooling is a down-sampling layer that applies region-wise operations on the data to extract a representative value for each region. Max-pooling is the most popular pooling technique and it finds the maximum value in its input feature map region. Most convolution networks include one or more fully connected layers at the end for final extraction of the answers.

Text convolution filters are usually one dimensional. The convolution network essentially identifies n-grams that are predictive at hand using varying filter sizes.

IV. Technical Approach

A. Dataset

The dataset is provided as a part of the Kaggle challenge - ’Quora Insincere Questions Classification’. Our solution kernel for this problem is hosted on github[1]. It consists of a labelled training set containing more than 1,300,000 examples and an unlabelled testing set with approximately 300,000 examples. The purpose of the unlabelled testing set is for the Kaggle team to evaluate competitors. Since unlabelled data is not useful for self evaluation, the testing set of this project is built by splitting the training set. A validation set if also extracted from the training set for cross-validation after epoch.The percentage of examples in each set are discussed in TABLE I.

TABLE I
Distribution of Examples

Training Set	***Validation Set***	***Testing Set***
56%	14%	30%

*Total examples: 200,000

[1]Source code available at https://github.com/slothie26/QuoraInsincereQuestionClassification

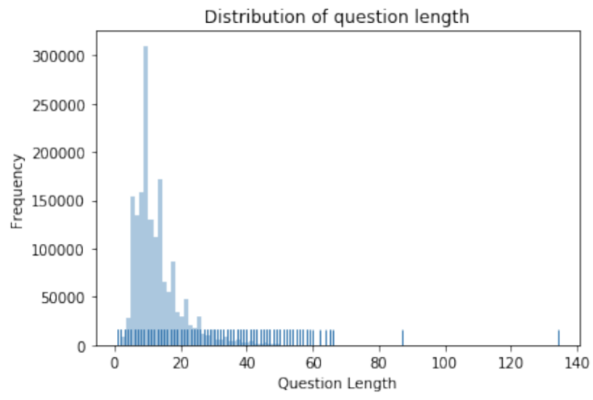
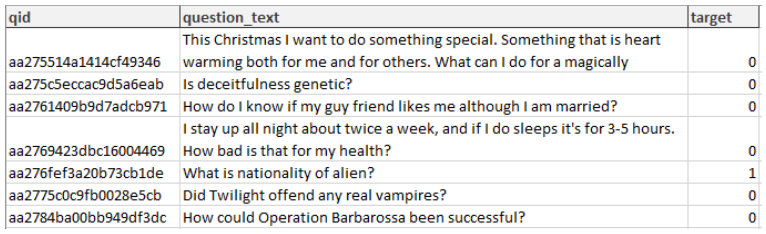


Fig. 2. Distribution of the length of each question in both the train and test sets. Min length = 1, max length = 134, average length = 12.8



qid	question_text	target
aa275514a1414cf49346	This Christmas I want to do something special. Something that is heart warming both for me and for others. What can I do for a magically	0
aa275c5eccac9d5a6eab	Is deceitfulness genetic?	0
aa2761409b9d7adcb971	How do I know if my guy friend likes me although I am married?	0
aa2769423dbc16004469	I stay up all night about twice a week, and if I do sleeps it's for 3-5 hours. How bad is that for my health?	0
aa276fef3a20b73cb1de	What is nationality of alien?	1
aa2775c0c9fb0028e5cb	Did Twilight offend any real vampires?	0
aa2784ba00bb949df3dc	How could Operation Barbarossa been successful?	0

Fig. 3. Sample from the Training Set

Each of the labelled examples has an associated unique id, the question, and the label associated with the questions. A sincere question is labelled as ’0’ and an insincere question is labelled as ’1’. The questions vary in size and a distribution of question lengths can be found in Figure 2. A sample of the training set can be seen is demonstrated in Figure 3

The data set is found to be quite imbalanced and is very large. The frequency of sincere and insincere questions is discussed in Figure 4. Due to a lack of extensive computational resources, a subset of this dataset was used as the training set which is further divided into testing and validation sets.

The subset used consists of 200,000 examples.

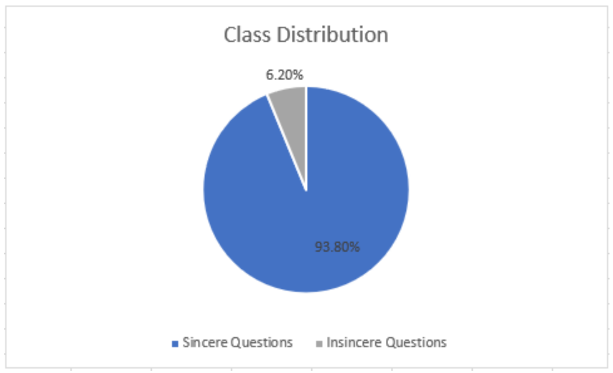


Fig. 4. Sample from the Training Set

B. Preprocessing

The data preprocessing approach that was undertaken, was an unconventional one which did not involve stemming or stopword removal. The motivation to follow this approach lies in the fact that the retention of this information helps neural network models to learn better.

The main goal in the preprocessing step is to obtain word embeddings that are as close as possible to the words present in the vocabulary. Constant monitoring followed by applied checks on the coverage of the vocabulary after calling each preprocessing function helped increase the utility of the embeddings as seen in Table II. The preprocessing functions included functions like purge_str, purge_num, obtain_misspellings and substitute_misspellings which removed all punctuations, numerals, got misspelled words present in questions and edited those misspellings respectively.

Pretrained word-embeddings are used because of their main advantage that involves no training phase. The embeddings chosen are pretrained on a part of the GoogleNews [11] word corpus which consists of $\sim 100,000,000,000$ words and contains 300-dimensional vectors for $\sim 3,000,000$ words and phrases. The Python package - gensim is used as our interface to word2vec. This is an open-source library for natural language processing and includes parallelized implementations of most word embeddings like FastText, word2vec and doc2vec algorithms.

This part of our machine learning pipeline is responsible for modeling the dataset in a form which could be useful to the training models in the next phases of the pipeline. A list of ordered dictionaries in Python is used to do this. Every question is read and split it into its constituent words. Every word is assigned with its corresponding embedding vector (each of dimension 300) and created an OrderedDict Python object for each question. Each OrderedDict object is then appended to a list to enable easy access of the embedded vector for each word in a given question. Finally, the representation looked like this:

```
[
  OrderedDict
      (What : ⟨ embedded vector ⟩,
      is : ⟨ embedded vector ⟩,
      hypothermia : ⟨ embedded vector ⟩,
      ? : ⟨ embedded vector ⟩),
  OrderedDict(...),
  ...]
```

TABLE II
COVERAGE METRICS FOR PREPROCESSING FUNCTIONS

Preprocessing Function	*Coverage wrt Text*
Original without preprocessing	78.75%
Removal of punctuation marks	89.98%
Removal of numerals	90.5%
Spell Check	98.95%

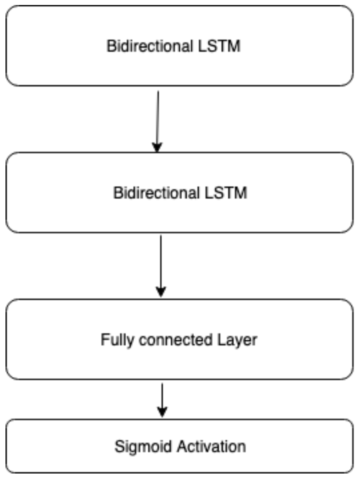


Fig. 5. BiLSTM architecture

C. Padding and Batch Generation

The data structure used is a list of ordered dictionaries, where each dictionary represented one question with keys as words and values as their corresponding vectors. Due to the large size of the dataset, it is not possible to load all the samples into the model at once. Thus, a batch generator function is created which sampled 128 samples every time to feed into the neural architecture. The models requires a fixed input shape, and since the questions are of variable length, the dimensions are also variable. To solve this problem, a threshold of 30 words per sentence is set. All the sentences with less than 30 words are padded with zeros while the sentences longer than 30 words are imputed at 30. This made all the inputs uniform, with the dimensions (30,300).

V. MODEL ARCHITECTURES

A. BiLSTM Model

LSTMs are a special kind of RNN, capable of learning long-term dependencies. LSTMs also have the same chain like structure, like the recurrent netwroks, but the repeating module has a different structure. Instead of having a single neural network layer, there are four, interacting in a very special way. An LSTM has three gates, to protect and control the cell state. The first step in an LSTM is to decide what information is going to be discarded from the cell state. The next step is to decide what new information is going to be stored in the cell state. This is done using two different layers of activation functions. After that the information about the old cell is dropped and the new information is added, as decided in the previous steps.Finally, the output is decided. This output is based on the cell state, but will be a filtered version. LSTMs are known to perform well for text classification problems but using a bi-directional LSTM tends to do better. This is because context is learnt both ways, in a forward as well as backward manner. Since our dataset involves questions in the form of sentences, having dependencies preserved from both sides.

In the experiments performed, the CuDNNLSTM layers from the python package Keras, are used to represent the LSTM layer. Two bidirectional LSTM layers are stacked one after the other specifying the input shape of each sample as (30,300), where 30 is the number of words in each sentence and 300 is the size of the embedding vector. A fully connected layer is added after the two bi-LSTM layers with the sigmoid activation function, to get the final output. The model computes a binary cross entropy loss since our aim is binary classification. The adam optimizer which is combination of the adagrad and momentum algorithms, is used to optimize this loss. The model is evaluated on the accuracy metric and run for 20 iterations, each with a step size of 1000. This implies that 1000 samples were subsampled from the entire data at each step. After the model is fit on the training data, cross-validation is performed, to avoid overfitting. After that the test accuracy is computed to see how many test samples are predicted correctly. The detailed architectural design is shown in Figure 5

B. BiGRU Model

Similar to a LSTM recurrent network, a Gated Recurrent Unit uses a set of weights to produce a hidden vector. Over a series of time-steps, it uses the same set of weights to combine the input at a particular time step with the previous hidden vector. However, unlike a standard Recurrent Neural Network, a GRU adds two "gates", an update gate, and a reset gate. GRU is very similar to an LSTM model, except that it has fewer parameters to learn and thus doesn't take as long to train. For our experimentation, the same model as the Bi-LSTM is used with the CuDNNLSTM layer replaced with the CuDNNGRU layer from Keras. The results obtained are very similar. Thus, a different model is tried with a more complex architecture to leverage the intricacies of the model. The architecture comprised of one bi-directional GRU layer followed by two fully connected layers with relu activation function. This is followed by batch normalization which is an optimization technique used to make the neural network less sensitive to initialization and has a higher learning rate. Dropout is applied which randomly leaves out certain samples from the training process. All these layers are finally passed into a fully connected layer at the end which produces the desired output. The Adam optimizer is used to optimize the binary cross entropy loss function and the model is computed with the accuracy metric. A detailed design is described in Figure 6

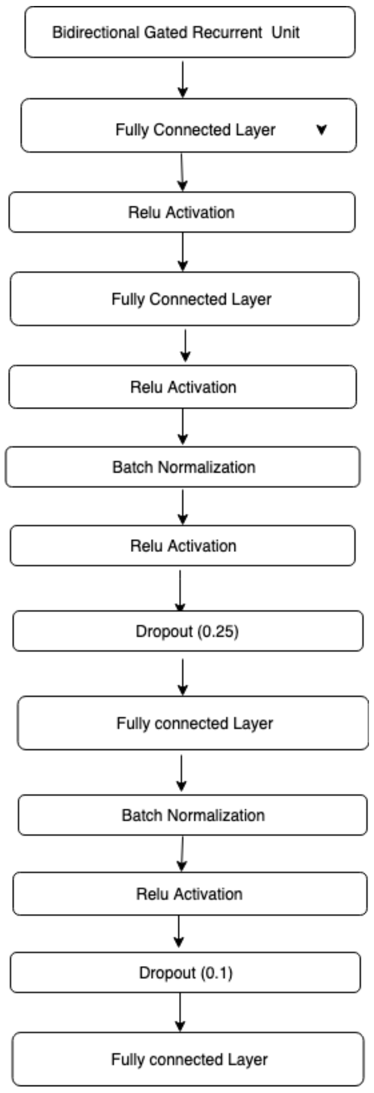


Fig. 6. Bi-GRU architecture

C. 1D Convolution Model

For the CNN implementation, four convolution layers are interleaved with dropout layers which selectively leaves out certain samples at each step. This is a regularization technique employed to prevent over-fitting of the data. Global Average pooling computes the average of all the values in the layer. Finally, it is passed through two fully connected layers at the end to get the final output. The input dimension is specified as (30,300) where 30 is the number of words and 300 is the size of the embeddings vector. The filter sizes are kept at 1,2,3 and 5 for each of the layers respectively. The Adam optimizer is used to optimize the binary cross entropy loss since its a binary classification problem. The model is computed for five epoch with 1000 steps, on the accuracy metric and evaluated accordingly. A detailed design implementation is shown in Figure 7.

VI. Results

A. Experimental Setup

1) System Requirements: The experiments are divided into two phases: Preprocessing and Model Training/Evaluation. The project is run on a compute engine on the Google Cloud platform. For the preprocessing and generation of embeddings a compute engine with a 52GB RAM to process the 1.3 million examples.

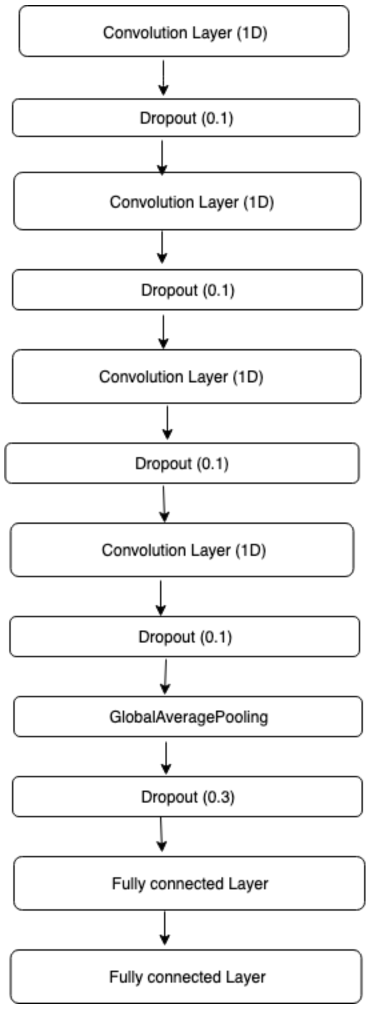


Fig. 7. CNN architecture

For model training and evaluation, the compute engine utilised an Inten Xeon 2.20 GHz, 8 core CPU, along with an NVIDIA Tesla K-80 GPU.

B. Evaluation Metrics

For the evaluation of the model, several different metrics are observed. Each of the observed metric has a different property it is evaluating the system for. The metrics that are used for the evaluation of this project are defined below:

1) **Accuracy**: Number of correct predictions made as a ratio of all predictions made [12]. It is the most commonly used metric but is better suitable if equal number of examples are present from each class.
2) **Binary Cross-Entropy Loss**: Also known as log loss. This metric evaluates the prediction of probabilities of membership assigned to a given class. [12] Predictions are punished or rewarded proportional to the confidence of the prediction depending on whether the prediction was correct or incorrect.
3) **Confusion Matrix**: Confusion matrix is a neat representation of the true positives, true negatives, false positives and false negatives. Evaluating the False positives and True negatives is especially useful in cases of unbalanced datasets.
4) **F1- Score**: This is the primary scoring metric in this project and also the suggested metric in the Kaggle Challenge. It is the harmonic average between the precision and recall [13].
Precision is defined as the number of correct positive results divided by the number of all positive results returned by the classifier, and recall is defined as the number of correct positive results divided by the number of all relevant samples.
5) **ROC Curve**: The receiver operating characteristic curve, or ROC curve, is a graphical plot that illustrates the diagnostic ability of a binary classifier system as its discrimination threshold is varied [14]. It plots the true positive rate against the false positive rate
6) **Area under Curve - ROC**: It represents the models ability to distinguish between positive and negative training examples[same reference as above].
7) **Precision Recall Curve**: These curves represent the tradeoff between true positive rate and the positive predictive value for a predictive model using different probability thresholds [15]. These are a better evaluation metric for imbalanced datasets. This metric is utilised in the case of binary classifications.

C. Model Evaluation

This project proposes the solution for the classification challenge using three different contrasting models as discussed earlier. The training, validation and testing accuracy's, binary cross entropy loss, F1 Score have been summarised in TABLE III.

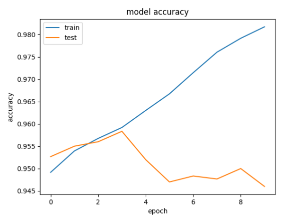
Figure 8 shows the Epoch vs Accuracy trends. In specific, the training and testing accuracies have been compared for each of the three models. The confusion matrices for the three models have been represented in Figure 9

The ROC Curves for the three models and their respective Area's under the curve have been shown in Figure 10. The precision recall curves for the three models have been compared with the baseline and summarised in Figure 11.

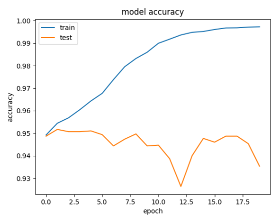
TABLE III
SUMMARY OF EVALUATION METRICS

Evaluation Metrics	**Models**		
	BiLSTM	***BiGRU***	***CNN***
F1 Score	0.5614	0.54	0.472
Testing Accuracy	94.79	94.16	94.43
Training Accuracy	98.17	99.73	97.93
Validation Accuracy	94.6	93.53	93.97
Training Loss	0.048	0.008	0.052
Validation Loss	0.182	0.432	0.222

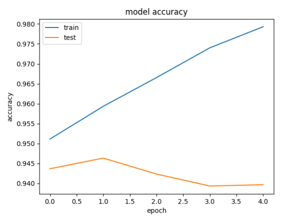
The recurrent neural network based models have performed significantly better than the convolution model as can be interpreted from F1 scores. The convolution model also took 4 times the training time than that of its counterpart recurrent



(a)

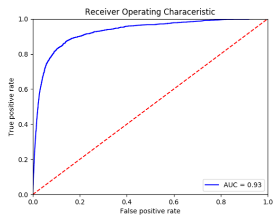


(b)

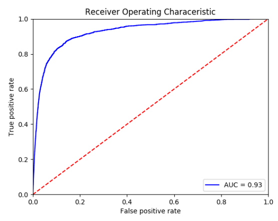


(c)

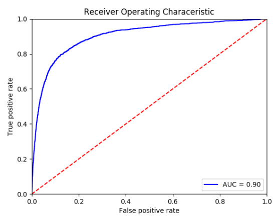
Fig. 8. Accuracy vs Epoch for (a)BiLSTM (b) BiGRU (c) CNN



(a)

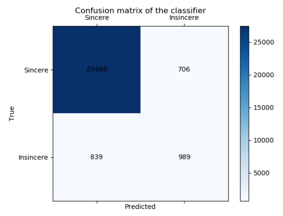


(b)

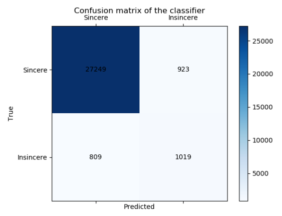


(c)

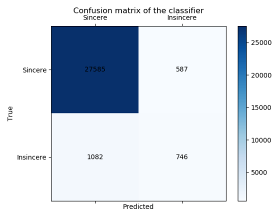
Fig. 10. ROC Curves for (a)BiLSTM (b) BiGRU (c) CNN



(a)

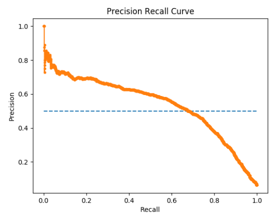


(b)

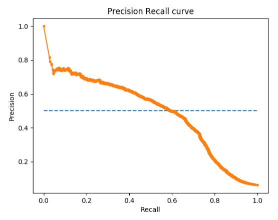


(c)

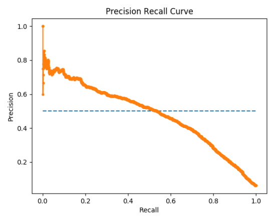
Fig. 9. Confusion Matrices for (a)BiLSTM (b) BiGRU (c) CNN



(a)



(b)



(c)

Fig. 11. Precision Recall Curves for (a)BiLSTM (b) BiGRU (c) CNN

models. Amongst the recurrent model, BiLSTM performs better than BiGRU in terms of both F1 score as well as Testing accuracy. But the difference between these models is not as significant as that with the convolution model.

The results can improve significantly more with access to better compute resources which would result in the complete dataset being used instead of a small subset that is currently being used in the project.

VII. Conclusion

The proposed project presents three contrasting models for detection of toxic content on Quora. All three models are finalised after several rounds of testing and improvement. Recurrent network models are observed to perform better than convolution models.

The results can be further improved by utilising the complete dataset. The overall project can improve by including a larger combination of pretrained embeddings, usage of attention layers, implementation of a deeper convolution networks.

Acknowledgment

The authors of this project would like to extend their thanks to Professor John Cho who is the instructor for the course CS249: Learning from Text 2019. The course covered most of the topics required for the completion of this project and the Professors insight and encouragement has led this project to successful completion.

The authors would also like to acknowledge the extensive dataset and the well-designed 'Quora Insincere Question Classification' challenge supported by Kaggle and Quora.

References

[1] Quora insincere questions classification. https://www.kaggle.com/c/quora-insincere-questions-classification/overview.

[2] Alex Wang and Vince Ranganathan. Is this question sincere? identifying insincere questions on quora using bert and variations. http://web.stanford.edu/class/cs224n/reports/custom/15763730.pdf, April 2019.

[3] Samuel Gabbard, Jinrui Yang, and Jingshi Liu. Quora insincere question classification.

[4] Alon Jacovi, Oren Sar Shalom, and Yoav Goldberg. Understanding convolutional neural networks for text classification. *CoRR*, abs/1809.08037, 2018.

[5] Alexis Conneau, Holger Schwenk, Loïc Barrault, and Yann LeCun. Very deep convolutional networks for natural language processing. *CoRR*, abs/1606.01781, 2016.

[6] Muhammad Zain Amin and Noman Nadeem. Convolutional neural network: Text classification model for open domain question answering system. *CoRR*, abs/1809.02479, 2018.

[7] Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. Recurrent neural network for text classification with multi-task learning. *CoRR*, abs/1605.05101, 2016.

[8] Peng Zhou, Zhenyu Qi, Suncong Zheng, Jiaming Xu, Hongyun Bao, and Bo Xu. Text classification improved by integrating bidirectional LSTM with two-dimensional max pooling. *CoRR*, abs/1611.06639, 2016.

[9] Peter J Angeline, Gregory M Saunders, and Jordan B Pollack. An evolutionary algorithm that constructs recurrent neural networks. *IEEE transactions on Neural Networks*, 5(1):54–65, 1994.

[10] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[11] word2vec. https://code.google.com/archive/p/word2vec/.

[12] Metrics to evaluate machine learning algorithms in python. https://machinelearningmastery.com/metrics-evaluate-machine-learning-algorithms-python/.

[13] F1 score. https://en.wikipedia.org/wiki/F1score.

[14] Receiver operating characteristic. https://en.wikipedia.org/wiki/Receiveroperatingcharac

[15] Roc curves and precision recall curves for classification in python. https://machinelearningmastery.com/roc-curves-and-precision-recall-curves-for-classification-in-python/.