Quora Insincere Questions Classification

Gowri Manasa Dharani, Keerthana Reddy Nerella
Texas A&M University
College Station
mdharani@tamu.edu, keerthana96@tamu.edu

Abstract

The internet can assist us in a plethora of tasks these days and one of those involves an online Question-Answer forum. Given the number of users using these forums, it is important for these forums to maintain reliable content and remove negative content. One such forum that we deal with in this paper is Quora where a user can ask a question related to any domain and anyone can post an answer to the questions. It is important to come up with an efficient way to achieve the goal of removing disruptive content and this is the task that this paper aims to deal with. Machine Learning models so far have been very efficiently dealing with text classification problems and we aim to implement models that can help solve this task efficiently. As a part of our study, we implemented three models: Logistic Regression, LSTM(Long short-term memory) and BERT(Bidirectional Encoder Representations from Transformers). We found that the BERT model exhibits the best performance.

Key words- Natural language processing, word embedding, LSTM, BERT

1. Introduction

1.1. Background

Quora can be seen as a very useful source of information for users interested in various domains and current events. It is used by different types of users belonging to all age groups like students, engineers, etc. Before the existence of such forums, people had to search on the internet or search in any physical material like books. Now, it is definitely useful to have an online forum where we can get ready made answers to questions. This is because most of the times, our questions must have occurred previously to someone else and someone must have provided the answer we are searching for.

1.2. Motivation

As in the case of any other online forum, there are a few users of quora who tend to misuse the platform. The feature of Quora that anyone can post questions and anyone can answer makes the task of filtering out toxic and insincere content very difficult. Also, manually analyzing every question and its responses is an arduous task and it is not possible because of the volume of content. As more and more people are posting content on the website, the data is continuously increasing enormously. So, it is important to come up with an efficient automated way to find out when insincere content is posted. To maintain the trust of its legitimate users, the disruptive content being posted on quora needs to be handled and removed as early as possible.

So, we have a challenge on the Kaggle website titled 'Quora Insincere Questions Classification'. This challenge defines the following classes of questions as insincere [1]

- Those which do not have a neutral tone and intend to serve as a comment on a particular group of people. Or it is rhetorical and serves as a declaration about a particular group of people.
- That try confirm or make a discriminatory remark on a group of people. They are based on strange hypothesis of a group of people. They might also criticize a characteristic which is incorrigible.
- Those which are based on false premises or false presumptions. Their grounds are not based on reality.
- They do not seek genuine answers and they have sexual content only for shock value.

Our task narrows down to classifying a question as sincere or not. This task is more difficult than it seems because of the backgrounds of the users, since there are diverse users from different parts of the world utilizing the platform. The usage of certain phrases and intentions might vary accordingly. Also, some words can be seen as discriminatory in one context but can be totally harmless in the other. We try to address these issues as much as possible.

1.3. Objective

The objective of this paper is to analyze the machine learning models pertaining to the task of text classification of questions on Qoura. We first analyze the available data set. Then, we include various pre-processing steps and observe how the results change as we vary the amount and type of pre-processing. Then, we start with building a baseline model and progressively, use a state of the art Natural Language Processing model and fine tune it to our task and then compare the results of different models.

2. Literature Survey

In our classification, the goal is to correctly find out if a question on quora is insincere or not. There is some amount of work done in this area previously. We will discuss the related work in this section.

The authors of [17] used multiple supervised learning models- Logistic regression, naive bayes, support vector machines and decision tree, random forest. For preprocessing, they removed stop words. Then, they used TF-IDF for text vectorization. Finally, they observed the maximum F1-score for Logistic regression model.

In [23], the authors used three variations of the Long short-term network along with different embeddings - LSTM, biderectional LSTM and LSTM+GRU. According to them, simple LSTM has given the highest F1 score.

The authors of [5] used TF-IDF, pre-trained glove embedding along with three different models- naive bayes, logistic regression and recurrent neural networks. According to them, the best model is word embedding with recurrent neural network model.

In [6], the authors have used only the training data set for the study. The used word embedding along with three models- multinomial naive bayes, k-nearest neighbor and logistic regression. They found that LSTM with GRU and achieved the best results.

The authors of [26], have also used recurrent neural network. They pointed out that deep learning techniques will give better performance than simple machine learning and data mining techniques. In [12] TF-IDF algorithm was used to find weight of words and then weight the vectors by TF-IDF value.

We can see that recurrent neural networks has been used extensively by most of the previous works in this area. According to [3], after training on a sufficiently large corpus, a model will perform better across many different data sets. We planned to use bidirectional encoder representation transformer BERT[3] model in this project and compare the results with that of recurrent neural networks because BERT is pre-trained on a huge corpus. The authors of [27] have performed conversational question answering using a model which used BERT.

From the analysis of previous works, we derive the following. Recurrent neural networks has been used widely to solve the problem. First, we build a baseline model using logistic regression, and then build the recurrent neural networks model. Finally, we will try the relatively new method BERT and find out if it outperforms the recurrent neural network's score.

3. Data set and preprocessing

3.1. Dataset

To achieve the purpose of accurately classifying the questions, Quora provided a huge labelled data set to Kaggle. They also provided word embeddings from GoogleNews, paragram, GloVe and wiki-news. The data set available to us from Kaggle consists of two parts, the training set and the testing set. The training data set contains three labels, the question ID, question text and the target. The target is 1 if the question is insincere, and 0 otherwise. The objective of this project is to learn from the training data set and make as accurate predictions as possible on the unlabeled test data set. To achieve the same, we first present the details about the work done so far in the form of baseline models, and further try a more advanced model to complete the prediction task. We built a simple bar chart to analyze

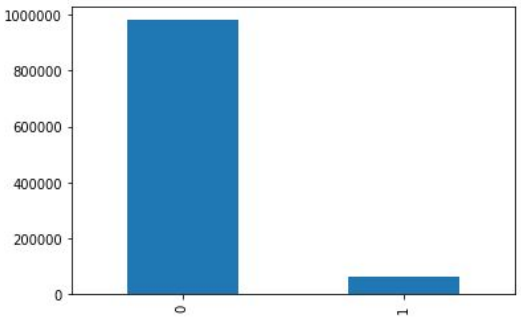

Figure 1. Distribution of training data

the data set. Upon inspection of the data set, it became evident that there are a lot more sincere questions, i.e, samples with label 1 than that of insincere questions. This is because of the obvious reason that most of the questions asked by users are genuine and there are a very few discriminatory comments or insincere questions. This can be identified as a class imbalance problem and the authors in [18] have pointed out that F1 score would be a better measure for prediction rather than accuracy.

4. Preprocessing

The tone used, sometimes a slang, some kind of jargon, abbreviations, misspelled words, word contractions, special characters are all nuances that could affect the task of pre-

diction if not properly taken care of. For this reason, we have identified preprocessing as an important step in this prediction task.

The preprocessing step is done in order to transform our samples into a form where the model we design for this task can perform better. This step ensures that all the data will be in same format. So, in later steps, performance of the machine learning models will be improved. Some important preprocessing steps are to delete invisible characters, correct misspelled words and to remove stop words [5]. Deleting invisible characters ensures that our models will not waste time on such characters. Correcting misspelled words helps in that the importance of any word is not left out. Stop words in english language are those which do not have much significance. That is, they do not add much meaning to the context. So, removing stop words reduces the number of words to process with little or no loss.[19]

A contraction is made by shortening and combining two words. We have substituted these contractions with the correct words. Also, we have removed all special characters and numeric characters from the samples as a part of cleaning up the text. We have observed an improvement in the results of the models after the incorporating the preprocessing steps.

5. Methodologies

5.1. Logistic Regression

5.1.1 TF-IDF (Term Frequency-Inverse Document Frequency)

Bag of Words is a algorithm that used the frequency of word occurrence on a document to represent the words. [21] has shown that TF-IDF is a simple but efficient way to represent words in a document based on their relevance to the query rather than in frequency. TF score captures how frequently a word occurs and the IDF score for the words that occur rarely is very high because these are the words that contribute to distinguishing the data. Calculating the inverse document frequency is important because the more a word occurs in a document, such as stop word like 'a', the less likely it is to contribute to the overall context of the document and therefore has less IDF score. It can also be said that this algorithm can be used to judge the topic of the document. Using this intuition, we implement a simple Logistic Regression model with TF-IDF word representation.

5.1.2 Implementation Details

After the preprocessing step, we initialize our vectorizer to TfidfVectorizer and we call fit_transform on the resulting training and testing set questions which converts raw document to a matrix containing tf-idf score. When initializing the vectorizer, we initialize the ngram from a range of 1 to 5 because the authors of [22] have shown that it is useful to use combinations of tokens can hold much more information than a single token. After this step, we proceed to perform logistic regression on this vectorized data with k-fold validation for k=10. To calculate the F1 score for this model, we use the f1_score method on the sklearn.metrics [7]. Also, we calculate the F1 score with a varying value of threshold starting from a value of 0.1 and upto the value 0.5. Since we are calculating the likelihood of a question being insincere by using logistic regression model, it is important to vary these thresholds which indicate that we vary the value at which the class is predicted as 1, i.e, question is predicted as insincere if the likelihood or probability turns out to be more than the threshold. We observe that we obtain the highest F1 score of 0.5932455512092998 at a threshold value of 0.17 and these results are consistent with the results published by [4].

5.2. LSTM+GRU

5.2.1 Word embeddings

After the preprocesing step, the next would be to find the correct representation for the words in our dataset. For the baseline model, we have used word embeddings to achieve this because word embedding technique allows us to have close representations for the words with similar meaning. This is done by representing a word as real-valued vector whose values are learned like a neural network. These vectors are calculated so that the vectors of words with similar meanings will be close. This is definitely a better representation than the traditional counting based methods to represent a word [11] because this model captures words based on their usage thereby judging how similar they are to each other and in turn capturing their meaning as well.

While TF-IDF gives a good representation of the data, we can use a better representation that not only capture frequencies and relevance in the document but also capture how similar two words are to each other, thereby digging into the documents and identifying context and also making analogical reasoning possible [16]. One such model that captures the word similarity and that can also be used calculating word vector representations is Word2Vec [15].

Word2Vec is a model that uses a two layer neural network which learns the context and linguistic content of the words. Unlike TF-IDF which produces a vector representation for the entire document, Word2Vec produces a vector representation for every unique word in the corpus. The vectors are so produced that words that have similar meaning or that are being used in the same context are located close to one another. The two flavours of this model, Continuous Bag-of-Words (CBOW) and skip-gram can predict the context of the target word given the neighbouring words or vice versa[15].

However, Word2Vec only uses the local context information and doesn't have any global information embedded in it. [20] proposed a model that combines local contextual information as well as global matrix factorization. This model, called GloVe creates a co-occurrence matrix by calculating the probability that a target word co-occurs with another word. The authors of [20] claim that this model produces a very meaningful vector space.

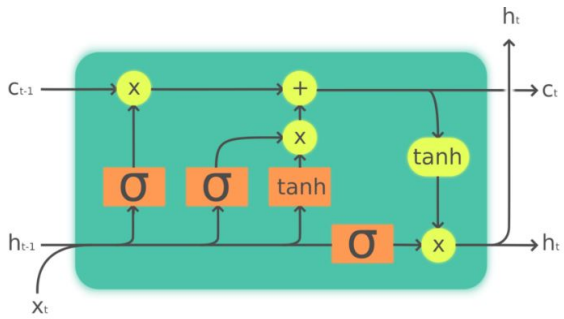

Figure 2. Long short term memory

5.2.2 LSTM

Recurrent neural networks have the ability to remember the information because of loops in the network. This is useful when we want to link previously known information to the task being performed. Recurrent neural networks are being used in various applications like language modelling, translation, text processing, speech recognition, etc. The problem with recurrent neural network is that though possible theoretically, practically, they cannot learn long-term dependencies.[8] Long short term memory network or LSTM is a special kind of recurrent neural network which gives better results than the standard recurrent neural networks.

LSTM is a kind of recurrent neural network that is capable of learning long-term dependencies. LSTM networks have a chain like structure in which each module has four neural network layers [25]. The LSTM consists of a sigmoid layer named as forget gate layer which decides how much information to send to the next level. The decision about what values to update is made by another sigmoid layer named as the input gate layer. Then, a vector of new values is created by a tanh layer. Finally, there will be another tanh layer followed by another sigmoid layer to decide which parts of the input to send out. Because of all these layers, an LSTM network is able to have a larger memory to remember information from time longer than that in normal recurrent neural network.

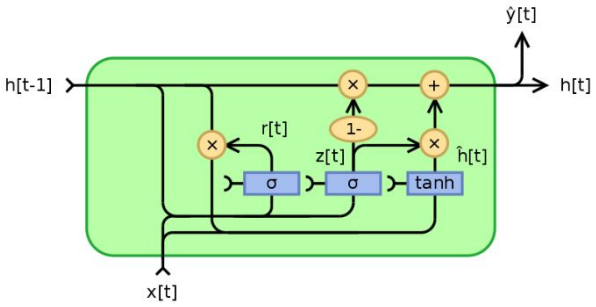

Figure 3. Gated Recurrent Unit

5.2.3 GRU

The Gated recurrent unit is an improved version of the recurrent neural network.[2] GRU is better than normal recurrent neural network in that it tackles the vanishing gradient problem.[10] This network has two layers. First layer is called as update layer and it is like forget and input layer of LSTM. It decides what amount of previous memory is to be retained and sent to next levels. The second layer is reset layer which decides how to integrate input from previous layers with the new input.

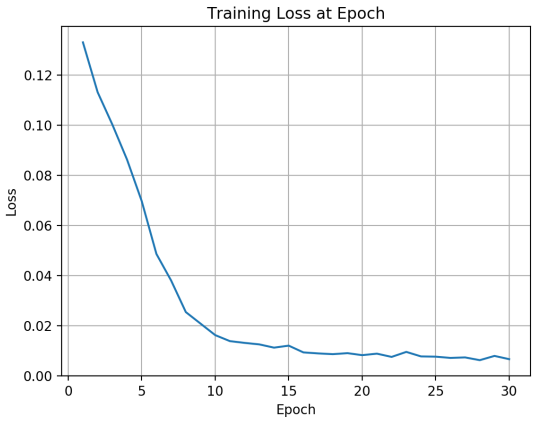

Figure 4. Training Loss for LSTM+GRU

5.2.4 Implementation Details

Here, for vectorizing the words of questions, we used Glove embedding. This gave better results when compared to that when google news, paragram or wiki news embeddings were used. We used Keras, which is an open source API to train and build deep learning models. We used the classes CuDNNLSTM and CuDNNGRU from keras.layers to implement the model. Using a combination of these two gave better results than each of them individually. These classes can be run only on GPU and are faster implementations when compared to LSTM and GRU. We wrap these with

bidirectional class so that context from forward and later will be taken into consideration. Then, we added a sigmoid activation function to the model. To measure the loss in the model, we used binary cross entropy error. Then, we used the adam optimizer and compiled the model. A similar model has been used in [5], where the authors have used a combination of LSTM and GRU. Finally, we fit the model using training data and validation data. Then, we test it using test data set. The metric used to measure the performance of the model is F1 score. We achieved F1 score of 0.606 at a threshold of 0.639.

5.3. BERT

Before BERT, in text processing, effort was put into looking at the text only in one direction or into combining the results from two directions. But, the important innovation in BERT is that it performs bidirectional training. BERT, Bidirectional encoder representations from transformers is a state-of-the-art machine learning model developed by the researchers at Google. This model is pretrained on the data from the entire wikipedia and a book corpus[3]. BERT can be used to perform various natural language processing tasks like question answering, and sentence classification.

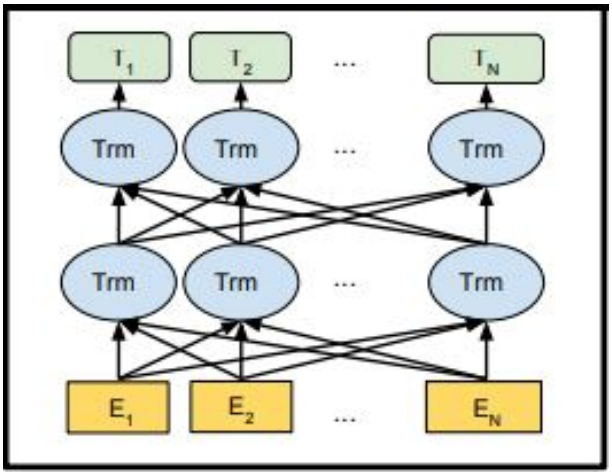

Figure 5. Bidirectional encoder representations from transformers

The concept of transformer in machine learning was introduced in the paper titled "Attention is all you need"[24]. The transformer architecture is advantageous in that it allows tasks to be taken care of in parallel. Transformer learns relationships related to context, between the words. The working of a transformer happens because of two parts- an encoder, which is responsible for reading the input and a decoder, which does the prediction. The encoder reads the complete input at once, instead of going from left-to-right or right-to-left. So, this architecture is used in the working of the BERT model.

According to [3], the biderectional training gives the model a higher sense of context and flow in the language. Training in BERT is done in two ways. In the first one called as masked LM, a percentage of words in a sequence are masked or hidden. The model then predicts these hidden words depending on the context provided by available words in the sequence. This is done by calculating probabilities using softmax function. The second way is next sentence prediction. Here, the model is given two sentences, and it has to predict if the second sentence occurs next to the first sequence. The probability whether next sentence is correct or not is calculated using softmax function.

5.3.1 Implementation Details

To implement this model, we use a combination of BERT and Fastai as mentioned in [9] and [13]. Fastai that is built on top of PyTorch requires the data to be passed in the form of a Databunch, so we wrap all our training and validation data in a wrapper and pass it on as a databunch to Fastai. After splitting our dataset into training and validation sets, we convert them into a databunch using the from_df method of TextDataBunch. This databunch needs a tokenizer and vocab to be passed as well.

Since fastai has its own tokenizer and vocab and we want to use BERT's word tokenizer and vocab, we need to customize the databunch to pass our own tokenozer and vocab. For this, we create our own tokenizer by loading the pretrained BERT model, bert-base-uncased and we wrap this tokenizer in our own tokenizer class. And finally to convert this tokenizer into a form that fastai can use, we wrap this in Tokenizer class and pass this customized tokenizer in the databunch. In a similar way, we wrap the vocab extracted from BERT pretrained model in Vocab and pass this in the databunch.

We now load the pretrained model using from_pretrained of BertForSequenceClassification. Finally, we wrap the customized databunch, a pre-defined loss function and the pretrained BERT model with Learner that handles everything that is required for training. To train the actual model, we use fit_one_cycle with a batch size of 1 and learning rate of 3e-5 as used in [3].

After training, the predicted labels are extracted by get_preds and we calculate accuracy and F1 score using accuracy_score and f1_score methods respectively and obtain an accuracy of 0.9626758541487224 and F1 score of 0.665614925577886.

6. Evaluation

For the purpose of correctly evaluating and comparing the performances of different models, we need a measure which takes into account all the possible aspects. A naive method is to use accuracy which is equal to the ratio of cor-

rectly classified questions to the total number of questions in the test data set. In our case, since the number of insincere questions is very less compared to the number of sincere questions, accuracy will be high, irrespective of whether the classification is correct or not. So, we cannot rely on accuracy to measure the performance of the models.

We will consider some other evaluation measures. The number of instances where the model correctly predicts the positive output is called as True Positive. If the model wrongly predicts the positive output, it is called as False Positive. Similarly, the number of instances where the model correctly predicts the negative output is called as False Positive and it is called as False Positive when it predicts negative output as positive. These metrics can be plotted in a confusion matrix as follows.

		Predicted Negative	Predicted Positive
Actual	Negative	TN	FP
Actual	Positive	FN	TP

$$precision = \frac{TruePositive}{TruePositive + FalsePositive}$$

$$recall = \frac{TruePositive}{TruePositive + FalseNegative}$$

$$F1score = 2\frac{precision * recall}{precision + recall}$$

Precision gives a measure of how accurate a model is according to the positive value. When false positive instance is costly, precision is preferred as a performance measure. Precision is the ratio of true positive to the sum of true positive and false positive. When a false negative instance is costly, the measure recall is preferred. Recall is the ratio of true positive to the sum of true positive and false negative. F1 score or F1 measure is a measure that takes into account both precision and recall. It is the harmonic mean of precision and recall. This measure will give us a balance between precision and recall. So, we use it because it is a perfect evaluation measure for our use case.

7. Results and Conclusion

According to the results, the BERT model has outperformed both the baseline models in terms F1, precision and recall scores. The Logistic Regression model achieved an F1 score of 0.593, while the LSTM+GRU model has achieved an F1 score of 0.606. The BERT model achieved an F1 score of 0.665. The plot of the F1 scores, precisions and recalls of these three models is given below.

To understand the meaning of a word in a text, a biderectional approach is better than traditional approaches. This

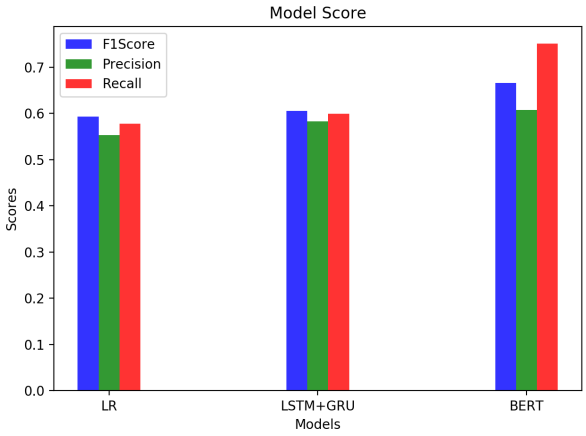

Figure 6. Results

is evident from our results because both the biderectional models LSTM+GRU and BERT are better than logistic regression. We have concluded that BERT performs much better than the other models because of it is a bidirectional model which uses transfer learning.

8. Future Work

We have used a state-of-the-art text classification model to achieve the results. But, recently another model called XLNet was designed by CMU and Google researchers in mid-2019. This model has outperformed BERT in 20 natural language processing tasks. BERT model uses special symbols during pre-training. These symbols are not present while fine-tuning the model to our task, and this results in a discrepancy between pre-training and fine-tuning. Another shortcoming of the BERT model is that when masking certain words, given the unmasked words, BERT assumes that all the masked words are independent of each other. The XLNet model overcomes these disadvantages of BERT. So, we predict that it might produce better results than BERT. We have tried implementing XLNet model to our problem by taking guidance from [14], but, due to lack of time and computational resources, we were not able to train the model completely.

9. Contributions

Gowri Manasa - Logistic Regression + TF-IDF(Baseline1), Using pretrained BERT model, Fitting model with databunch, label prediction.

Keerthana Reddy - LSTM + GRU(Baseline2), Customizable BERT Tokenizers, Vocab, Wrappers for Fastai (Fastai integration).

References

[1] Quora insincere questions classification. [online], 2019. https://www.kaggle.com/c/quora-insincere-questions-classification/data. 1

[2] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014. 4

[3] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. 2, 5

[4] S. Gabbard, J. Yang, and J. Liu. Quora insincere question classification. 3

[5] S. Gabbard, J. Yang, and J. Liu. Quora insincerequestion classification. 2019. 2, 3, 5

[6] B. Gaire, B. Rijal, D. Gautam, S. Sharma, and N. Lamichhane. Insincere question classification using deep learning. 2

[7] K. Ganesan. Build your first text classifier in python with logistic regression. 3

[8] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 4

[9] A. Jha. Fastai integration with bert: Multi-label text classification identifying toxicity in texts. 04 2019. 5

[10] S. Kostadinov. Understanding gru networks. 4

[11] O. Levy, Y. Goldberg, and I. Dagan. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225, 2015. 3

[12] L. Li, L. Xiao, N. Wang, G. Yang, and J. Zhang. Text classification method based on convolution neural network. In *2017 3rd IEEE International Conference on Computer and Communications (ICCC)*, pages 1985–1989. IEEE, 2017. 2

[13] C. McCormick and N. Ryan. Bert fine-tuning tutorial with pytorch. 04 2019. 5

[14] C. McCormick and N. Ryan. Xlnet fine-tuning tutorial with pytorch. 2019. 6

[15] T. Mikolov, G. Corrado, K. Chen, and J. Dean. Efficient estimation of word representations in vector space. pages 1–12, 01 2013. 3

[16] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. *Advances in Neural Information Processing Systems*, 26, 10 2013. 3

[17] A. Mungekar, N. Parab, P. Nima, and S. Pereira. Quora insincere questions classification. 2

[18] P. Nima. Quora insincere questions classification. 04 2019. 2

[19] D. M. Olga Davydova. Text preprocessing in python: Steps, tools, and examples. 3

[20] J. Pennington, R. Socher, and C. Manning. Glove: Global vectors for word representation. volume 14, pages 1532–1543, 01 2014. 4

[21] J. Ramos. Using tf-idf to determine word relevance in document queries. 01 2003. 3

[22] M. Shirakawa, T. Hara, and S. Nishio. N-gram idf: A global term weighting scheme based on information distance. pages 960–970, 05 2015. 3

[23] M. SINGH. *QUORA INSINCERE QUESTIONS CLASSIFICATION*. PhD thesis, 2019. 2

[24] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017. 5

[25] S. Yan. Understanding lstm and its diagrams. 4

[26] D. Yogeshwaran and N. Yuvaraj. Text classification using recurrent neural network in quora. 2019. 2

[27] C. Zhu, M. Zeng, and X. Huang. Sdnet: Contextualized attention-based deep network for conversational question answering. *arXiv preprint arXiv:1812.03593*, 2018. 2