ABSTRACT

The objective of this project was to estimate whether a particular Quora question is insincere or not. Here Quora is a Community Question and Answer platform which enables people to learn about certain topics from each other. The data provided by Kaggle contains two datasets which are train.csv and test.csv along with four word embeddings which are glove.840B.300d (Glove), GoogleNews-vectors-negative300 (Google), paragram_300_sl999 (Paragram) and wiki-news-300d-1M (Wikipedia). The train.csv and test.csv include 1306122 and 375806 samples respectively. Furthermore, train.csv contains three features that are qid, and question_text along with target and test.csv contains two features that are qid along with question_text.

The Methodology of the project is divided into three parts. In the first part of the project, dataset train.csv and test.csv were loaded and were tokenized using Spacy Model (en_core_web_trf) with the help of Pytorch. Then the tokenized sentences were encoded by replacing the words with Unique Word IDs. The Second Part of the Project was to load the above-mentioned four word embeddings only for unique words in all the tokenized sentences. Furthermore, four word embeddings were averaged to create a fifth word embedding (Average Word Embedding). The third part of the project involved the creation of a Deep Neural Network (DNN) model which was inspired by the model made by Shofiqul & Ngahzaifa [2022] and had two main components that are Bidirectional Long Short-Term Model (BiLSTM) and the Bidirectional Gated Recurrent Unit (BiGRU). In the DNN model, the output of BiLSTM is fed to the input of BiGRU and the output of both BiLSTM and BiGRU are concatenated after passing them through individual Global Maximum Pooling Layer. Moreover, there were five such models built for five different word embeddings. In the end, the predicted output from all five models which be ensembled by averaging them.

Based on the training along with the validation results of the five models the top two models are the Glove and Average models. The F-Score of the Glove Model was 96.61 % and the Average Model was 96.43 % on the training data. Furthermore, the validation accuracy of Glove and Google Model was 96.07 % and 96.12 % respectively.

1 INTRODUCTION

The project involves the classification of Quora (Community Question and Answer Service) questions into sincere and insincere questions. Kaggle organized this project and provided the necessary dataset which includes train.csv and test.csv along with four word embeddings. Here, Kaggle is a Machine Learning (ML) and Artificial Intelligence (AI) competition platform. The four word embeddings given were glove.840B.300d (Glove), GoogleNews-vectors-negative300 (Google), paragram_300_sl999 (Paragram) and wiki-news-300d-1M (Wikipedia) and all these word embeddings have a dimension of 300. Another word embedding was generated by averaging the above four word embeddings and this word embedding is called Average word Embedding.

The Glove pre-trained word embeddings were developed by utilizing an unsupervised learning Algorithm known as Global Vectors (GloVe) [5]. Similarly, the Google pre-trained word embeddings were generated by the word2vec tool made by Google which efficiently implements skip-gram architectures and continuous bag-of-words [4]. However, the Paragram pre-trained word embeddings were finely tuned on the SimLex999 dataset with the help of the Paraphrase Database (PPDB) [6,7]. Likewise, Wikipedia pre-trained word embeddings were trained on Wikipedia 2017, statmt.org news dataset and UMBC web base corpus [2].

The train.csv had 1306122 samples and three features which were qid, question_text and target. Furthermore, test.csv had 375806 samples and two features which were qid and question_text. In the above-mentioned features of train.csv and test.csv, the qid column contains unique identifiers of each question of Quora, the question_text contains the Quora Question, and the target column contains value 1 for insincere question and value 0 for a sincere question.

To solve the problem, the project was divided into three parts that are Data Preprocessing, Word Embedding Refinement and Neural Network Model development. In the Data Preprocessing part, the question sentences in question_text are first tokenized using Spacy Model (en_core_web_trf). Then, the tokenized sentences will be encoded by replacing tokens with Unique Word IDs. These encoded tokens will be the input to the Neural Network Model. Now, in the Word Embedding Refinement Part, the word embeddings of only those words were loaded that were present in the question_text column for all five word embeddings (i.e. Glove, Google, Paragram, Wikipedia and Average). Lastly, in the Neural Network Model development part, a model with BiLSTM and BiGRU as main components was built in which the output of BiLSTM was sent to the input of BiGRU [11]. The train.csv data was split in such a way that 80 % of the data was used to train the Neural Network Model and 20 % of the data was used to validate the model.

2 METHODOLOGY

This section is divided into four parts which are Data Preprocessing, Word Embedding Refinement, Neural Network Model development and Threshold Detection.

2.1 Data Preprocessing

One of the most significant parts of this project was to clean the data that was provided by the Kaggle. The Data Preprocessing section is divided into three parts namely Question Tokenization, Question Encoding and Padding.

In Question Tokenization, the questions present in the question_text column of train.csv and test.csv were tokenized by splitting the tokens according to the Spacy Model en_core_web_trf which is an English transformer pipeline (roberta-base). These tokenized questions were kept in a new column called question_text_nlp. After this, all unique tokens were extracted from all the tokenized questions in the question_text_nlp column. Each of these unique tokens was given a unique word identifier (IDs) that starts from 1. These unique IDs were utilized in the Question Encoding Part.

In Question Encoding, the tokenized questions in the question_text_nlp column of train.csv and test.csv were converted to numbers by replacing each of the tokens with their unique IDs which were defined above. These encoded sentences were stored in a new column called question_text_encode. After this, the longest sentence in the question_text column of train.csv and test.csv was extracted and the number of tokens in this sentence was stored in maximum_sentence_length. This number will be used in the Padding Part.

In Padding, the length of all the encoded sentences in the column question_text_encode in train.csv and test.csv was transformed to the length stored in the variable maximum_sentence_length. This particular task allows the encoded questions to be processed by the Neural Network model.

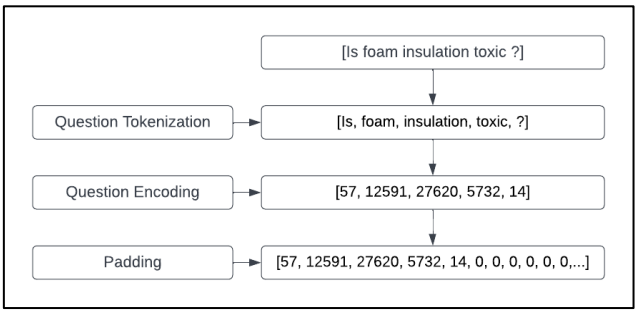

Figure 1: Different Stages in the Data Preprocessing of the Quora Insincere Question Classification Project

2.2 Word Embedding Refinement

Another important part of the Quora Insincere Classification problem was to load and curate word embeddings. There were four word embeddings that were provided by the Kaggle. These word embeddings were glove.840B.300d (Glove), GoogleNews-vectors-negative300 (Google), paragram_300_sl999 (Paragram) and wiki-news-300d-1M (Wikipedia) and all these word embeddings have a dimension of 300. The Word Embedding Refinement is divided into two parts that are Word Embedding Filtering, and Average Word Embedding Generation.

In the Word Embedding Filtering section, first, the unique words or tokens from the question_text_nlp column in train.csv and test.csv are extracted and stored in a unique word table. Then, word embeddings of only those words are loaded which are stored in the unique word table. Those unique words which do not have a word embedding in Glove or Google or Paragram or Wikipedia are provided with a vector of zeros with a dimension of 300. Then the word embeddings of these unique words were arranged or sorted according to the unique word IDs generated in the above Data Preprocessing section.

In the Average Word Embedding Generation section, a fifth word embedding was generated by averaging the above four filtered word embeddings (Glove, Google, Paragram and Wikipedia).

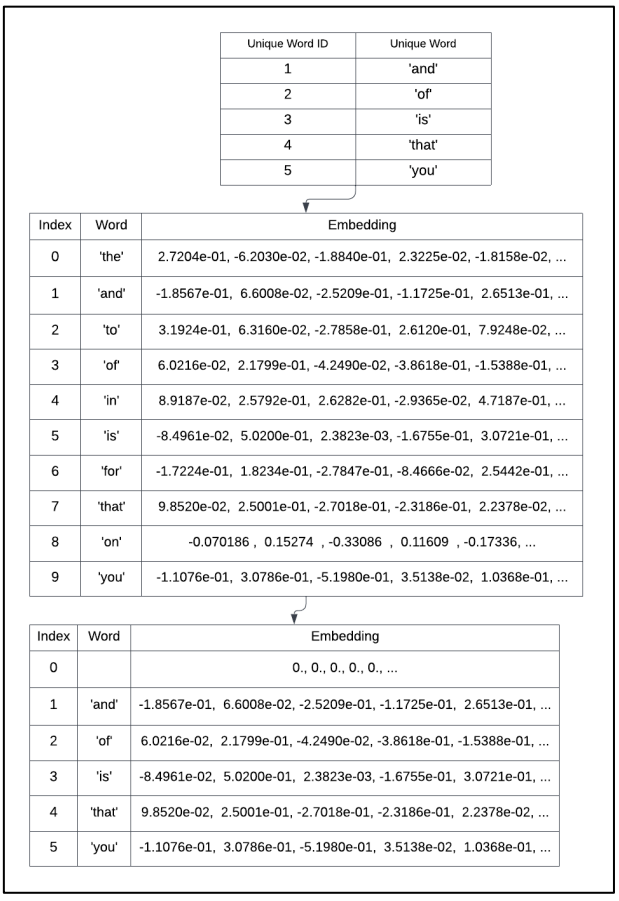

Figure 2: Different Steps in the Word Embedding Refinement (Filtration) of the Quora Insincere Question Classification Project

2.3 Neural Network Model

The most important aspect of any project is the development of a model which can predict values for certain input features. Similarly, in the Quora Insincere Question Classification Project, a Neural Network Model was built which had two main components. These two components were the Bidirectional Long Short-Term Memory (BiLSTM) Model and the Bidirectional Gate Recurrent Unit (BiGRU) Model. This Neural Network Model is inspired by the Model developed by Shofiqul & Ngahzaifa [2022]. Also, this model has six variations based on five word embeddings (Glove, Google, Paragram and Wikipedia) and the sixth variation was formed by combining (ensemble) the outputs of the above five word embeddings. Moreover, there were nine other layers that helped in the prediction of the target column in train.csv and test.csv. These layers are explained below:

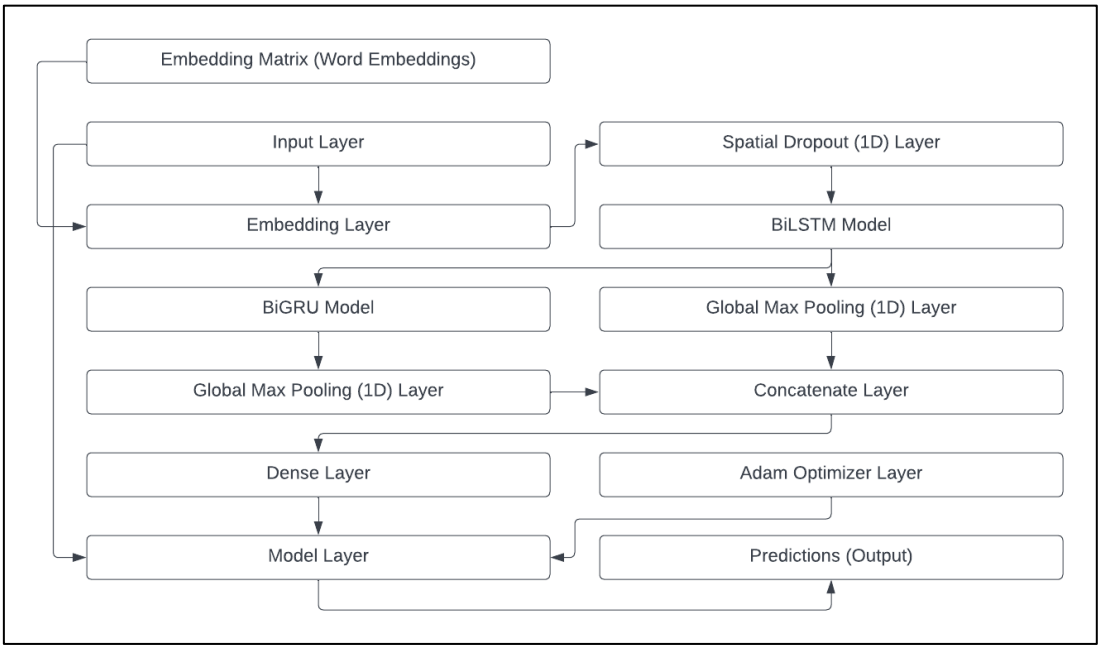

Figure 3: Different Layers of the Neural Network Model and the connection between different Layers of the Model

2.3.1Input Layer.

The question_text_encode column present in train.csv and test.csv will be first fed to the Input Layer which is utilized to instantiate tensors. These tensors flow across all the Neural Network Layers and they have a shape similar to that shape of the training and testing data. After initiation of the tensors, these tensors are sent to the Embedding Layer.

2.3.2Embedding Layer.

The Embedding Matix i.e., the Word Embeddings (Glove, Google, Paragram and Wikipedia) were utilized layer in this layer. The tensors received in this layer were from the Input Layer which contains the question_text_encode column in tensor format.

The main function of the Embedding layer is to map the encoded question words (that are integer numbers) to the vectors (word embeddings) found at the corresponding index in the embedding matrix, for instance, the sequence [1, 2]

would be converted to [embeddings[1], embeddings[2]]. This means that the Embedding layer output will be a three-dimensional tensor of shape (samples, sequence_length, embedding_dim) [3].

2.3.3Spatial Dropout (1D) Layer.

The Spatial Dropout Layer takes the output of the Embedding Layer as Input to provide regularization. It drops an entire 1D (one-dimensional) feature map only when the adjacent pixels in the data (feature map) are highly correlational. This helps increase independence between feature maps [12].

2.3.4BiLSTM Model.

The BiLSTM is made from two LSTM models arranged in opposite directions. LSTM stores input for a longer period of time and has three gates namely input, forget and output gates. BiLSTM models are really helpful when the input context is required, and sentiment classification has to be performed. The output of BiLSTM is sent to the BiGRU Model and to the Global Max Pooling (1D) Layer [11].

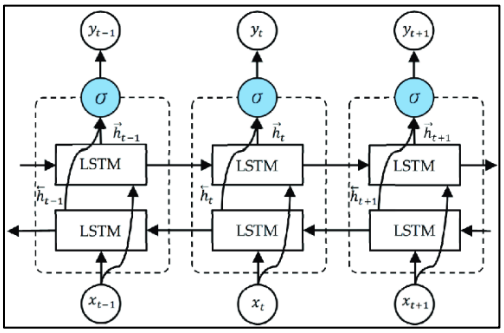

Figure 4: BiLSTM Model

2.3.5BiGRU Model.

Similar to BiLSTM, BiGRU is made from two GRU models arranged in opposite directions. The input to BiGRU is the output of BiLSTM. GRU has the capability to store memory from previous detections and has two gates namely reset and update gates. The output of BiGRU is sent to the Global Max Pooling (1D) Layer [11].

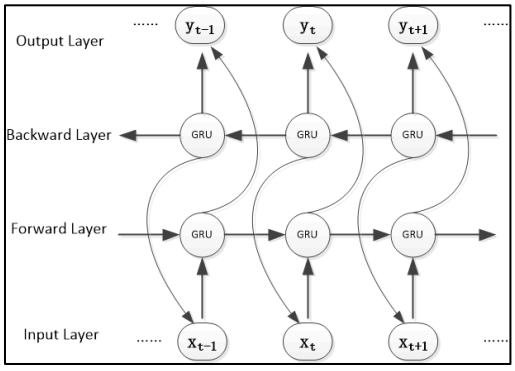

Figure 5: BiGRU Model

2.3.6Global Max Pooling (1D) Layer.

The Quora Project Neural Network Model has two Global Max Pooling (1D) Layer. Out of these layers, one receives input from BiLSTM and the other receives input from BiGRU. Global Max Pooling Layer retrieves the maximum value over the time dimension and therefore helps in down sampling the input representation.

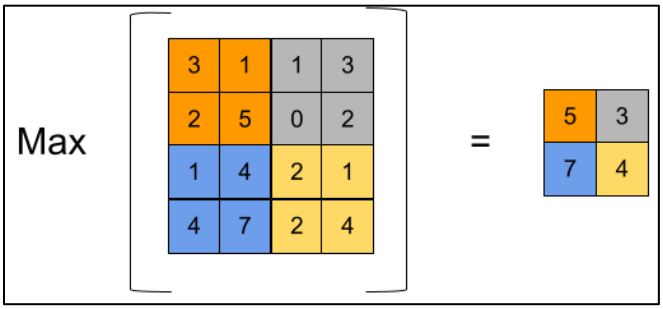

Figure 6: Max Pooling Process Explanation

2.3.7Concatenate Layer.

As the name suggests the Concatenate Layer is utilized to join outputs of two Global Max Pooling (1D) Layers. Here the output of the Global Max Pooling (1D) Layer is a one-dimensional vector [8].

2.3.8Dense Layer.

The Dense Layer receives input from the Concatenate Layer. The Dense Layer performs the following operation: output = activation(dot(input, kernel)) where the dot represents the dot product between input and kernel, activation is the element-wise activation function which is sigmoid for the Neural Network Model, and the kernel is a weights matrix created by the layer [9].

2.3.9Adam Optimizer Layer.

Adam is an adaptive learning rate optimization algorithm which is utilized in the training part of deep neural networks. It utilizes predictions of first and second gradient moments to adjust or adapt the learning rate.

2.3.10 Model Layer

The Model Layer groups all the above layers into a single object. Furthermore, the input and output paths of the Neural Network Model are also defined using the Model Layer [10].

2.4 Threshold Detection

The output from the Model Layer of the Neural Network Model will be float values between 0 and 1. Therefore, to get proper target values we need an output of either 0 or 1. To achieve this receiver operating characteristic (ROC) curve was used with kneedle algorithm [1]. The ROC curve uses the actual target values with predicted target values to plot a curve with false-positive rates on the x-axis and true positive rates on the y-axis for different threshold values. Then kneedle algorithm is used to find the knee point of the ROC curve which is a point with a specific false positive rate and true positive rate. The Threshold value for this knee false positive rate and the true positive rate is extracted and is utilized to convert values between 0 and 1 to either 0 or 1.

3 RESULTS AND FINDINGS

There were five variations of the above-mentioned Neural Network Models based on five word embeddings (Glove, Google, Paragram, Wikipedia and Average) and the sixth variation of the neural network model is formed by combining (ensemble) the predicted targets of the above five models. The Precision, Recall and F-Score of six variations of the Model on Training Data are provided in Table 1 which is given below.

Table 1: Precision, Recall and F-Score of different Neural Network Models on Training Data

Model Variation	Precision	Recall	F-Score
Glove	96.76 %	96.75 %	96.75 %
Google	96.51 %	96.32 %	96.40 %
Paragram	96.22 %	96.35 %	96.28 %
Wikipedia	96.12 %	96.42 %	96.17 %
Average	96.61 %	96.69 %	96.64 %
Ensemble	96.72 %	95.41 %	95.84 %

Table 2: Accuracy and Loss of different Neural Network Models (average of 3 epochs) on Training Data

Model Variation	Accuracy	Loss
Glove	95.96 %	10.17 %
Google	95.81 %	10.65 %
Paragram	95.43 %	11.89 %
Wikipedia	95.77 %	10.78 %
Average	95.82 %	10.56 %

Table 3: Accuracy and Loss of different Neural Network Models (average of 3 epochs) on Validation Data

Model Variation	Accuracy	Loss
Glove	96.02 %	9.91 %
Google	95.94 %	10.22 %
Paragram	95.46 %	11.54 %
Wikipedia	95.99 %	10.12 %
Average	96.00 %	10.05 %

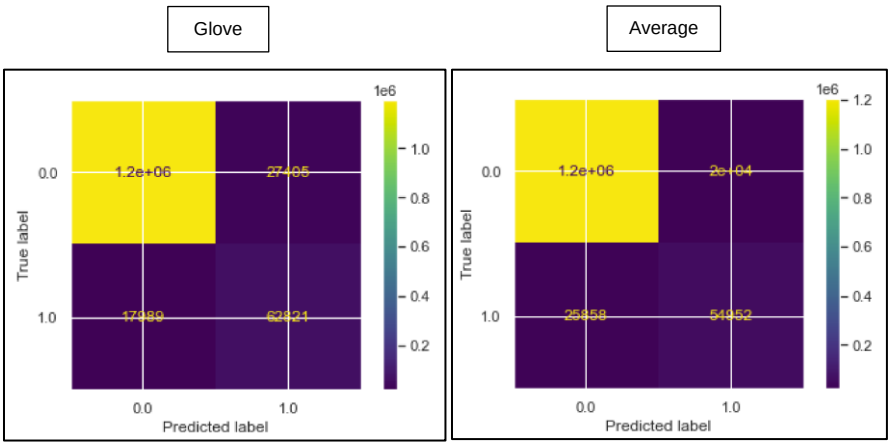

Figure 7: Confusion Matrix of Glove and Average Neural Network Model

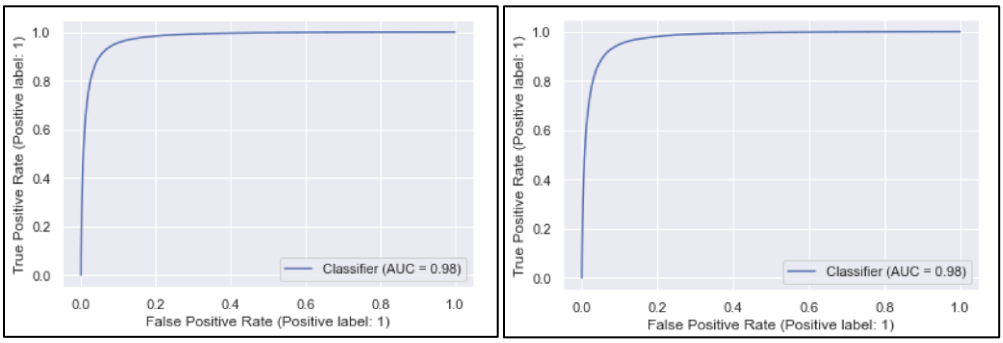

Figure 8: ROC curve of Glove and Average Neural Network Model

The Accuracy and Loss of five variations (Glove, Google, Paragram, Wikipedia and Average) of the Neural Network Model on Training and Validation are given in Table 2 and Table 3. The Training and Validation Data were formed by splitting the question_text_encode column in train.csv into 80 % for training data and 20 % for validation data.

Furthermore, the confusion matrix and ROC curves of the Glove and Average Model are shown in Figure 7 and Figure 8 because these were the best models due to their higher F-Score (96.75 % and 96.64 % respectively for training data) and Accuracy (95.96 % and 95.82 % respectively for training data along with 96.02 % and 96.00 % respectively for validation data).

4 DISCUSSIONS AND CONCLUSIONS

Almost all of the models have shown similar performance on training data and validation data, however, the Ensemble Neural Network model has shown the least Recall (95.41 %) and least F-Score (95.84 %). Out of all six models, the best model was Neural Network Model with Glove Word Embedding as it showed an Accuracy of 95.96 % for training data and 96.02 % for validation data which was the highest compared to other models. Furthermore, the Loss of Glove Neural Network Model was 10.17 % for training data and 9.91 % for validation data which is the lowest when compared to other Models.

Data Preprocessing played a significant role in improving the accuracy of the Neural Network Model by tokenizing and encoding large data sets (train.csv and test.csv). Furthermore, the accuracy can be better improved by using the Precision-Recall curve to find a better Threshold for converting target values between 0 and 1 to either 0 or 1.

The Quora Insincere Classification Project opens various conversations for future improvements. The Word Embeddings of each word can be multiplied by their Inverse Document Frequency (IDF) Score to make less frequent words more important. Moreover, sentence embeddings can be generated by averaging the word embeddings for words present in a sentence. The Word Embeddings in a Sentence Embedding can also be multiplied by IDF to improve Accuracy. These sentence embeddings can be directly fed to the Input Layer of the Neural Network Model. This can potentially improve Accuracy.

REFERENCES

[1] Arvkevi. Arvkevi/kneed: Knee point detection in python. Retrieved July 3, 2022 from https://github.com/arvkevi/kneed

[2] Fasttext Team. English word vectors · fasttext. Retrieved July 3, 2022 from https://fasttext.cc/docs/en/english-vectors.html

[3] Francois Chollet. The keras blog. Retrieved July 3, 2022 from https://blog.keras.io/using-pre-trained-word-embeddings-in-a-keras-model.html

[4] Google Team. Google code archive - long-term storage for google code project hosting. Retrieved July 3, 2022 from https://code.google.com/archive/p/word2vec/

[5] Jeffrey Pennington. Retrieved July 3, 2022 from https://nlp.stanford.edu/projects/glove/

[6] John Wieting, Mohit Bansal, Kevin Gimpel, Karen Livescu, and Dan Roth. 2015. From paraphrase database to Compositional Paraphrase model and back. (August 2015). Retrieved July 3, 2022 from http://arxiv.org/abs/1506.03487

[7] John Wieting. Retrieved July 3, 2022 from https://www.cs.cmu.edu/~jwieting/

[8] Keras Team. Keras Documentation: Concatenate layer. Retrieved July 3, 2022 from https://keras.io/api/layers/merging_layers/concatenate/

[9] Keras Team. Keras documentation: Dense layer. Retrieved July 3, 2022 from https://keras.io/api/layers/core_layers/dense/

[10] Keras Team. Keras Documentation: The model class. Retrieved July 3, 2022 from https://keras.io/api/models/model/

[11] Md. Shofiqul Islam and Ngahzaifa Ab Ghani. 1970. A novel BiGRUBiLSTM model for multilevel sentiment analysis using deep neural network with BIGRU-BILSTM. (January 1970). Retrieved July 3, 2022 from https://link.springer.com/chapter/10.1007/978-981-33-4597-3_37

[12] Tensorflow Team. Tf.keras.layers.spatialdropout2d : Tensorflow Core v2.9.1. Retrieved July 3, 2022 from https://tinyurl.com/2tuvut3x