The School of Mathematics

THE UNIVERSITY *of* EDINBURGH

LSTM for identifying the duplicates of Quora question pairs

by

Cong Wang, s1919768

August 2019

Supervised by
Dr Daniel Paulin
Dr Victor Elvira

Own Work Declaration

Contents

Executive summary

Quora is a question answering website where users can ask some questions and responders will answer, on these websites there are some duplicate questions. Under this circumstance, responders tend not to answer the same questions and some users may not have access to some high-quality answers that have been given to the same questions with those users ask. As a result, it is necessary to identify duplicate questions and build some relationship between them.

The data given is a training dataset and a test dataset and in each record in these two dataset contains a question pair and the label representing if these two questions are duplicate. The method that I use is the Siamese neural network which uses the same layers and same weights for question pairs. The key point of the neural network is LSTM layer for two inputs after implementing word2vec embeddings. Once going out from LSTM layer, two inputs are merged and pass through Dropout, BatchNormalization and Dense layer. The output is the probability that the inputted questions are duplicate.

Once the model is done, the quality of the model is also evaluated. Considering the output is the probability of being duplicate, I choose a threshold and if probability is greater than it then the questions are declared as duplicate. Then the recall-precision curve is plotted f-value which is the trade-off between them is computed. The maximum of f-value is around 0.8 and at the same time the threshold is set at 0.38. What's more, ROC is also plotted and the best AUC is 0.910, which verifying the model is desirable.

1 Introduction

1.1 Background

Record linkage (RL), also named entity resolution (ER) is the process of identifying similar records that may represent the same entity (for example, a product) from one or several data sources[1]. It is especially necessary as the size of database grows dramatically. It is of great value for integrating multi-source data to improve data quality and achieve better data discovery. This also allows for less efforts and costs in data capture for further research. Record linkage has been applied to many aspects, for example deleting duplicate records, linking business records and matching bibliographic and e-commerce products.
Quora is a website where users can ask questions and others will respond to these questions[2]. Answers of high quality are up-voted and they provide valuable information for some topics. However, as we all know, on such question answering websites repeated questions are common and such circumstances are also unavoidable for Quora, especially as the size of questions grows. If treated independently, duplicate questions may result in a phenomenon that users who ask duplicate questions tend to fail to get high-quality answers that already exist because responders may unwilling to answer the same question twice. It is helpful to predict duplicate questions in that it contribute to lead users to the same question that they want to ask directly and prevent responders always answering same questions and reduce their burden. As a result, it improve the overall user experience.

1.2 Introduction of datasets

Quora question pairs dataset: Question pairs along with labels indicating whether the two questions contain the same meaning or not.

1.3 Goal

The goal of this project is to train a machine learning algorithm using the training samples to predict whether two questions are duplicates of each other (i.e. have the same meaning), and test this algorithm on the test samples.

2 Explanatory data analysis

2.1 Overview of datasets

The Quora question pairs dataset is composed of a training dataset of 323,480 question pairs and a test dataset of 80,871 question pairs.
Each record has the following fields:

- id - the id of a training set question pair
- qid1, qid2 - unique ids of each question (only available in train.csv)
- question1, question2 - the full text of each question
- is_duplicate - the target variable, set to 1 if question1 and question2 have essentially the same meaning, and 0 otherwise.

In training dataset, two records of question2 are missing and they are dropped because they only account for a small proportion. Of the remaining 323,478 training question pairs, there are 203,851 (63.02%) samples having a negative (0) label and 119,627 (36.98%) samples having a positive (1) label, making our training dataset unbalanced.
Even though each question pair is unique, there are 9,530 (1.50%) questions out of 634,113 unique questions appearing more than once. The most frequent question appears 43 times. The

number of times a question appears against the number of questions for these occurrences is shown in figure 1.

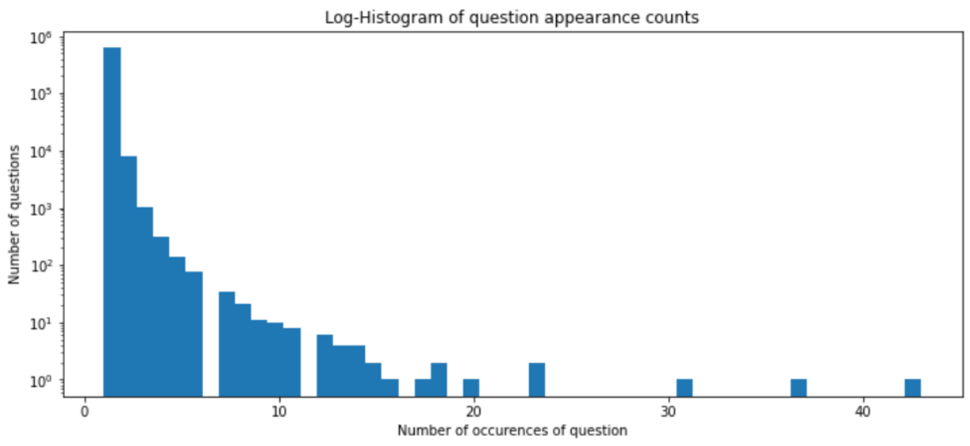
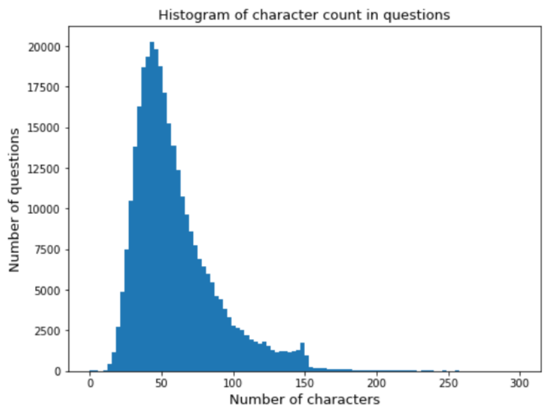


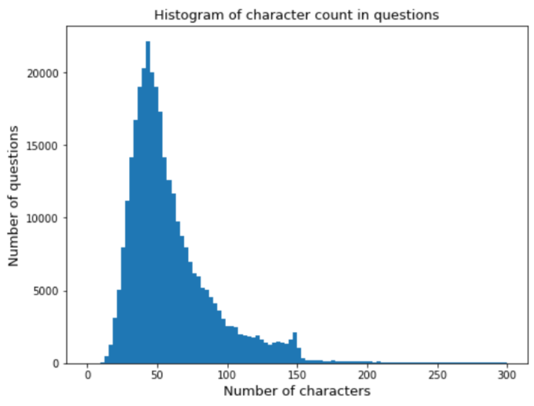
Figure 1: Question appearance counts.

Observationally, the character set of training dataset is not strictly ASCII and some questions include non-ASCII characters.

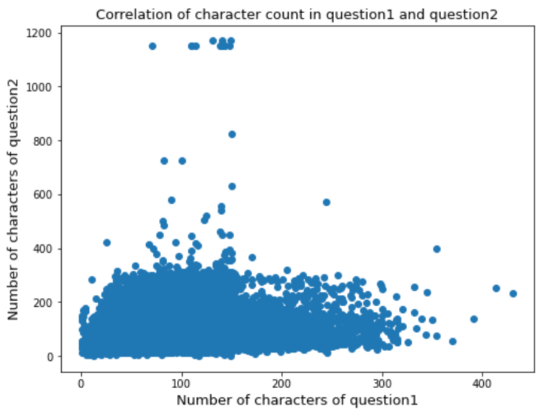
2.2 Feature explanatory data analysis



(a) Question 1



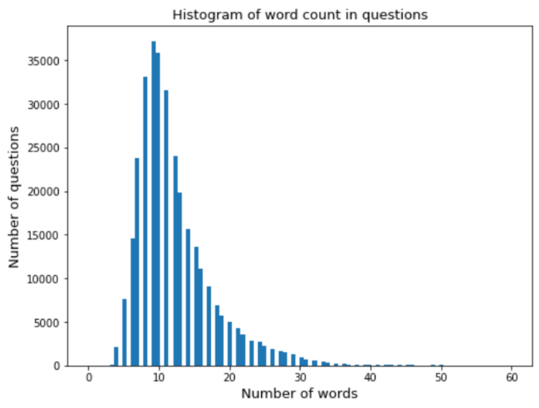
(b) Question 2



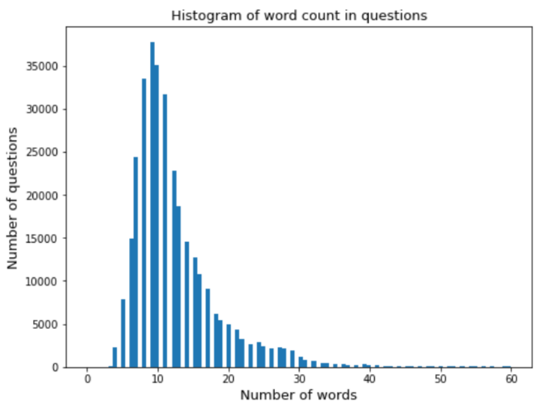
(c) Question 1 vs Question 2

Figure 2: Character counts in questions.

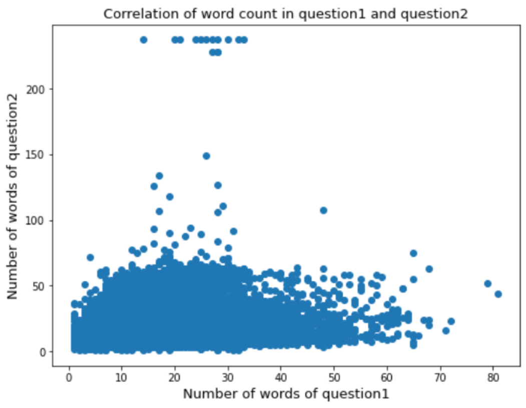
The length of questions vary a lot from questions to questions and the minimum value of the length of question1 and question2 is 1 and the maximum value is 430 and 1,169 for question1 and question2 respectively. Figure 2 shows the length of questions against the number of questions for these sizes. As we can see, the size of majority of questions ranges from 15 to 150 and the histograms are truncated at 300 characters. One thing that catches my attention is the steep cut-off around 150 characters.
Apart from that, number of words of each question is different. Figure 3 shows the number of words of questions against the number of questions for these numbers. Majority of questions contains about 10 words and there exists no huge difference between question1 and question2. The minimum value of the number of words of question1 and question2 is 1 and the maximum value is 80 and 245 for question1 and question2 respectively. On top of that, the mode of the distribution of the number of words is similar to that of the size of questions.



(a) Question 1



(b) Question 2



(c) Question 1 vs Question 2

Figure 3: Word counts in questions.

It is observed that for question pairs there exist some common words in question1 and question2. What's more, hypothesis is that duplicate questions are more likely to have a higher proportion in word share. Figure 4 shows the distribution of the percentage of word in common between question1 and question2 when the questions are duplicate or not. As we can see, duplicate questions tend to have more common words compared with different questions, which is consistent with our hypothesis.

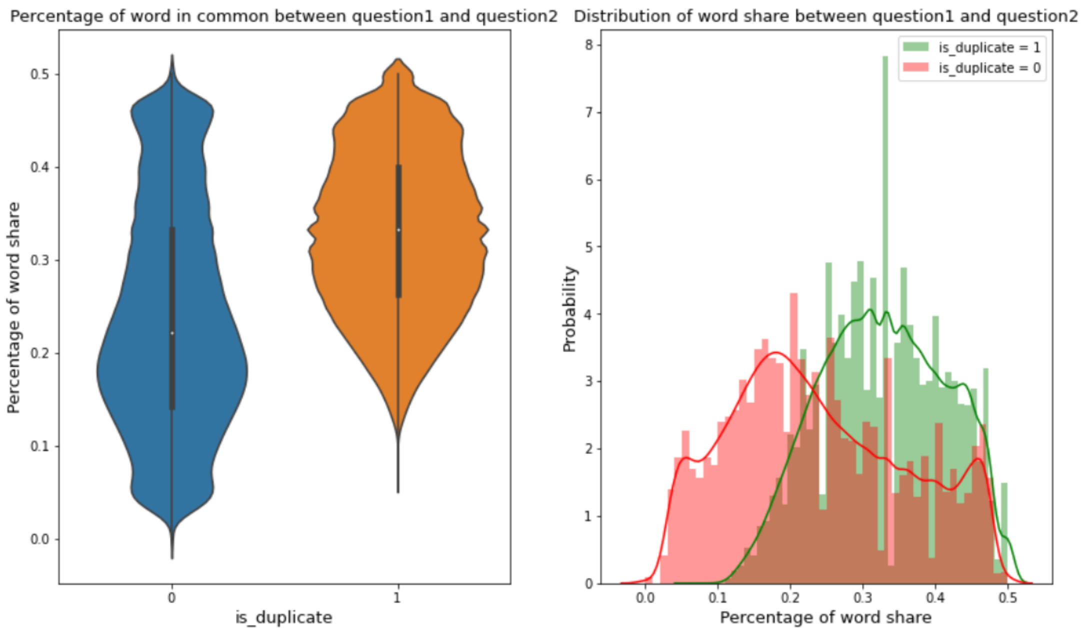


Figure 4: Percentage of word in common between question1 and question2.

2.3 Preprocessing

The questions are mixed with upper case characters and lower case characters. Therefore, all characters are converted to lower case characters. Besides, it is necessary to clean the questions in that the format of some character add some noise to comparison between questions. What comes first is that characters in the brackets are deleted. Besides, some abbreviations are converted to normal form, for example, 'what's' is converted to 'what is', 'n't' is converted to 'not' and etc. What's more, some punctuations (e.g. comma) are deleted.

3 Modeling

3.1 Overview of the model

There is a network that plays an extremely wonderful role in predicting the similarity between two different systems, Siamese Network. Siamese neural network is an artificial neural network that use the same weights for two different input vectors to compute output vectors[3]. It is widely used to checking the similarity between two images and two texts. Therefore, Siamese Network is a excellent choice for our model.
For the inputted two different questions, they are passed to LSTM layer at first. After that, they are merged and go into Dropout layer to prevent overfitting[4]. Then they are passed to BatchNormalization layer to improve performance, speed and stability of neural network. On top of that, they go into the most fundamental layer, Dense layer. After that, they are passed to Dropout layer and BatchNormalization layer in sequence to improve the quality of this neural network. Finally, they go into Dense layer and generate the output vector.
Figure 5 shows overview of the process of my neural network and the details will be covered in the following section.

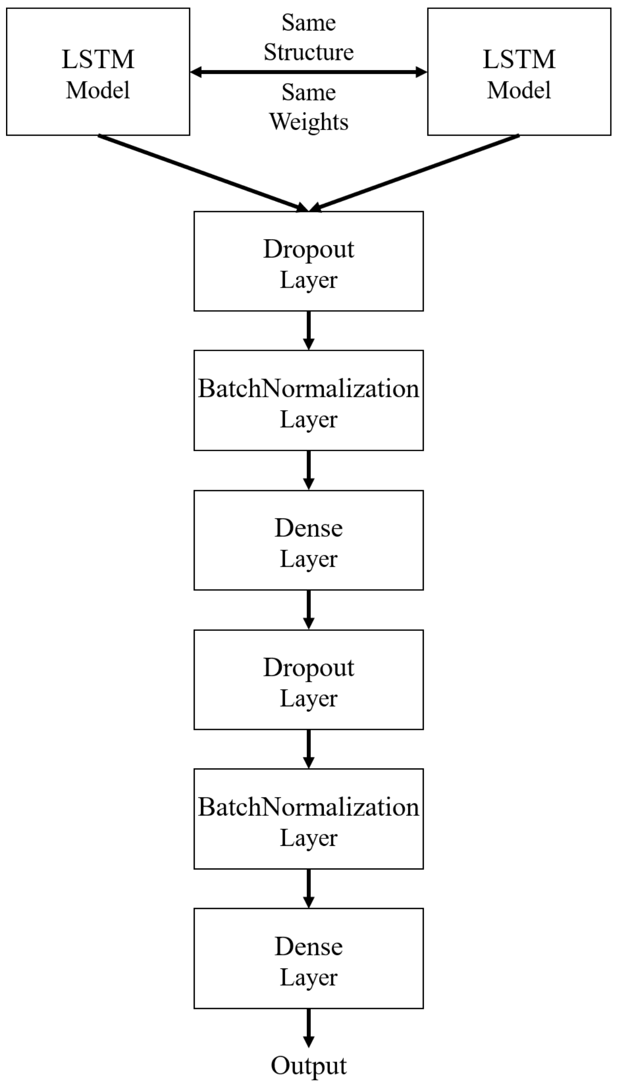


Figure 5: Overview of the neural network.

3.2 Vectorizing the inputs

As we all know, none of machine learning models can cope with texts like a human. However, they can work well by mapping the statistical structure of the texts, which has been applied to many tasks including classification of documents, sentiment analysis, question answering systems and so on. Three major different ways of dealing with texts is to segment texts into words, segment texts into characters, group n consecutive words or characters together respectively and then transform them into vectors.

In my neural network, the size of training dataset is huge and segmenting texts into characters will pose more burden to the model. Therefore, segmenting texts into words is sufficient for our network. One thing that deserves our attention is that while implementing tokenization we are supposed to take not only training dataset but also test dataset into account. If only texts of the training dataset are tokenized, the words in the test dataset are neglected so that the result tend to fail to meet our expectation. In order to limit the size of the model, only the most 200,000 frequent words are taken into account. Fortunately, only 85,536 unique words are found in the training dataset and test dataset. After tokenization, texts are truncated at the first 30 words in that the number of words of most of questions is less than 30 words.

In the following step, the task is to transform each word into a vector. The traditional method is one-hot encoding, which is the most common and basic method. However, the vectors obtained from one-hot encoding are high-dimensional and sparse. A substitute for this method is word

embeddings, which can generate low-dimensional and dense vectors. Besides, the vectors are in float type. There exist two different ways to get word embeddings, one is learning word embeddings by associating a vector with a word at random. The problem resulting from this method is that the final embedding space is of no structure. For example, sometimes word 'approximate' and 'around' have the same meaning but they end with totally different vectors. Another method is use pretrained word embeddings and it is the method that is taken for my neural network. Considering the Word2vec algorithm is the most popular and successful method, finally it is selected for embedding layer.
After downloading Word2vec embeddings and importing it, I found that it contains 300-dimensional vectors for 3,000,000 words. Then an embedding matrix is required for embedding layer and the first argument of its shape should be min(max_words, unique_words), which is 85,536 in my neural network. The second dimension of the shape is the dimension of the embedding layers, which is 300 in my model. Then use the vectors of the words from Word2vec for the words in our questions if they can be found in Word2vec.

3.3 Network layers

3.3.1 LSTM layers

After passing through embedding layer, two different inputs go into Long Short-Term Memory (LSTM) layer, which is a type of recurrent layers. The advantage of LSTM layer is that it deals with sequences by iterating through them and keeping record of previous states[5]. It is helpful for vanishing gradient problem and the network with a high depth of layers. Figure 6 shows the theory of LSTM layer.

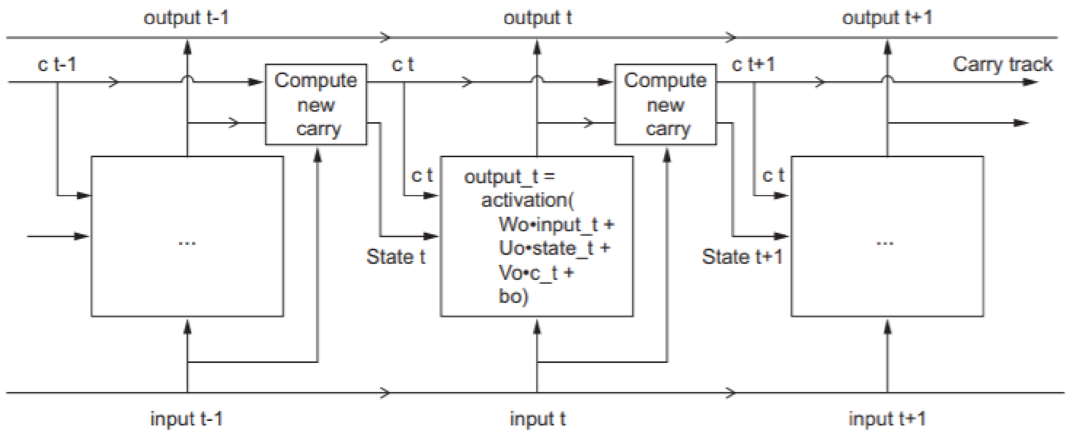


Figure 6: Theory of LSTM layer.

Compared to traditional recurrent layers, an extra data flow is added to the model. The values of it is C_t at different time, where C represents carry. It is combined with input and recurrent connection via a dense transformation. The pseudocode details of the LSTM architecture is as follows.

$$\begin{aligned} output_t &= activation(dot(state_t, U_o) + dot(input_t, W_o) + dot(C_t, V_o) + b_o) \\ i_t &= activation(dot(state_t, U_i) + dot(input_t, W_i) + b_i) \\ f_t &= activation(dot(state_t, U_f) + dot(input_t, W_f) + b_f) \\ k_t &= activation(dot(state_t, U_k) + dot(input_t, W_k) + b_k) \\ c_{t+1} &= i_t * k_t + c_t * f_t \end{aligned} \tag{3.1}$$

From the equation, we can see that additional data flow have an effect on the next timestep. The value of C at next timestep is computed from three different sources.

As I mentioned above, siamese neural network is used for my model. Though two inputs are passed to two LSTM layers separately, the weights of these two LSTM layers are the same.

3.3.2 Dropout layer

Dropout layer, just as its name implies, means dropping out some units in the neural network. However, rather than actually dropping these units, it just chooses units at random during the training process and ignores them temporarily. What's more, each individual node may be dropped with a probability of p or be kept with a probability of 1-p. Figure 7 shows the theory of the Dropout layer.

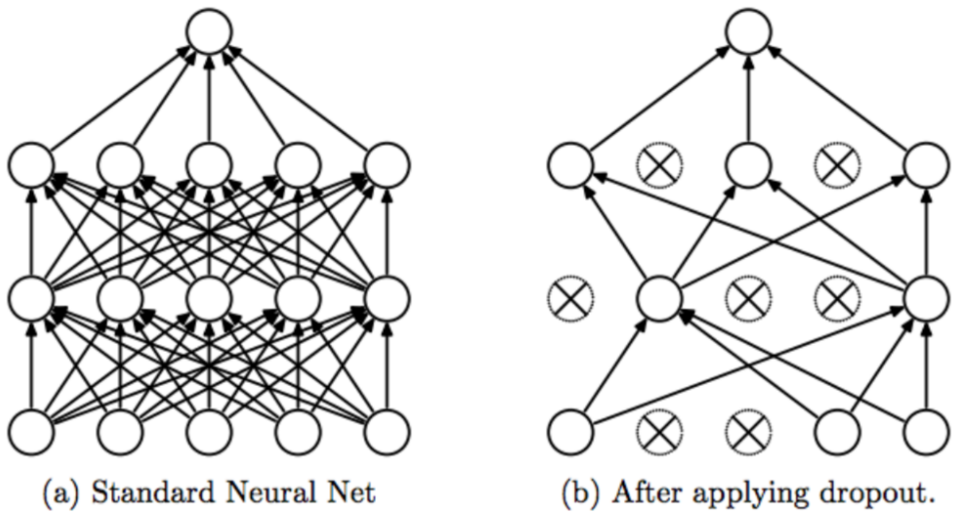


Figure 7: Theory of dropout layer.

As we all know, regularization is a way of preventing overfitting by adding penalty to loss function in traditional machine learning. By regularization the model does not learn interdependent set of feature weights. A fully connected layer occupies majority of the parameters so that neurons become dependent on each other which suppresses individual power thus results in overfitting. Dropout is an approach of preventing overfitting in neural network.

3.3.3 BatchNormalization layer

After passing through Dropout layer, the merged vectors go into BatchNormalization layer, which normalizes the input by scaling the activations. Besides, it allows every layer of the neural network learn by themselves and independent from other layers. It improves the stability of the neural network by subtracting the mean value of the batch which is the output of the last activation layer and dividing by the standard deviation of it. Denote values from a mini-batch by $x_i, x_2, ..., x_m$, then equation 3.2 shows the process of normalisation.

$$\begin{aligned} \mu &= \frac{1}{m}\sum_{i=1}^{m} x_i \\ \sigma^2 &= \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu)^2 \\ \hat{x_i} &= \frac{x_i - \mu}{\sqrt{\sigma^2 + \epsilon}} \end{aligned} \tag{3.2}$$

Then BatchNormalization layer adds two trainable parameters and the output can be computed from equation 3.3.

$$y_i = \gamma \hat{x_i} + \beta \tag{3.3}$$

where γ is the standard deviation parameter and β is the mean parameter.
Thus, BatchNormalization makes Stochastic gradient descent (SGD) implement the denormalization by changing only these two parameters rather than changing all the weights in order to keep the stability of the neural network.
Apart from that, BatchNormalization layer also has slight regularization effect. Similar to the Dropout layer that was mentioned above, BatchNormalization layer adds some noise to each hidden layer's activations. As a result, it can curb overfitting to some extent. However, its function is not superior to that of Dropout layer and it is always combined with Dropout layer for application.
In conclusion, BatchNormalization layer increases the quality of the neural network by improving the speed, stability and performance.

3.3.4 Dense layer

Dense layer is the most popular and the most basic layer in neural network. The intermediate dense layer uses 'relu' (rectified linear unit) as activation function, which works well in the situation that the input data is vectors and the output is categories (0s and 1s). It is a function which takes bigger value of the input data and zero, which is shown in figure 8.

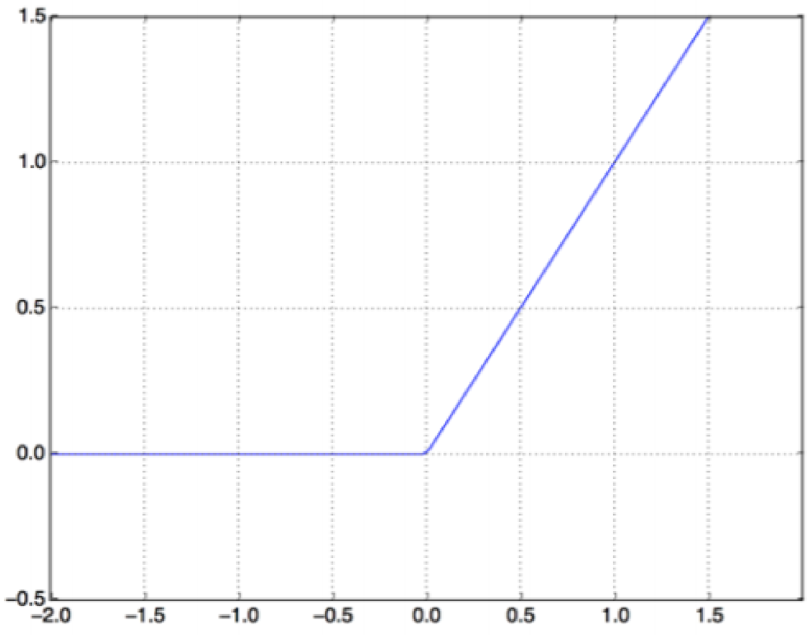


Figure 8: The rectified linear unit function.

The parameter of Dense layer is the number of hidden units of the layer. A hidden unit is a dimension in the representation space of the layer. Dense layer with a relu activation function implements tensor operation like equation 3.4.

$$output = relu(dot(W, input) + b) \tag{3.4}$$

where W is the weight matrix with shape (dimension of input, number of hidden units). The product of the input data and the weight matrix project the input data onto a space whose dimensions is equal to the number of hidden units. b is the bias vector which is added to the product.
The final dense layer uses 'sigmoid' as its activation function.The output of this function is a score between 0 and 1 which stands for the probability that if the inputted two questions are duplicates of each other. Figure 9 shows the 'sigmoid' function.

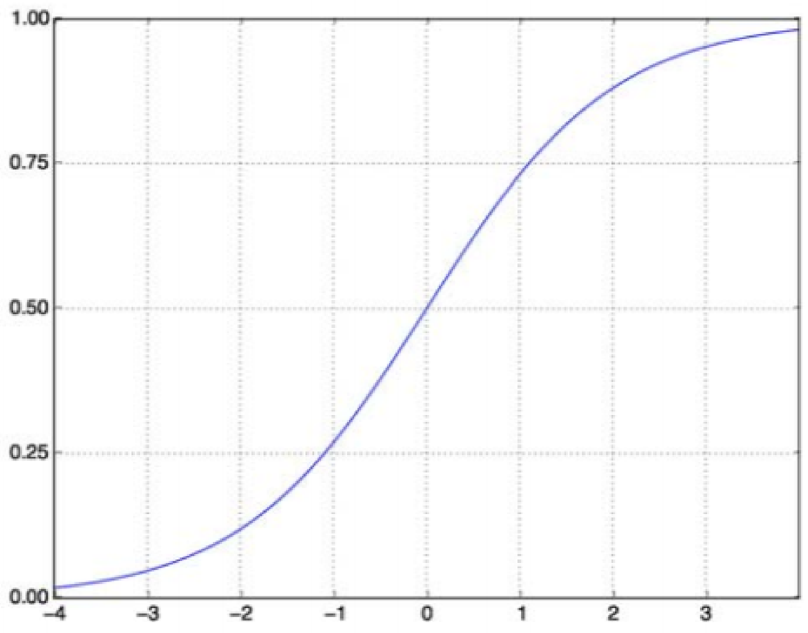


Figure 9: The sigmoid function.

3.4 Preparation for the training data and validation data

In order to evaluate the quality of our model and prevent overfitting, the training dataset is supposed to be splitted into training data and validation data. In many cases, K-fold cross-validation is implemented to reduce the variance of the validation scores. Figure 10 shows the theory of 3-fold cross-validation.

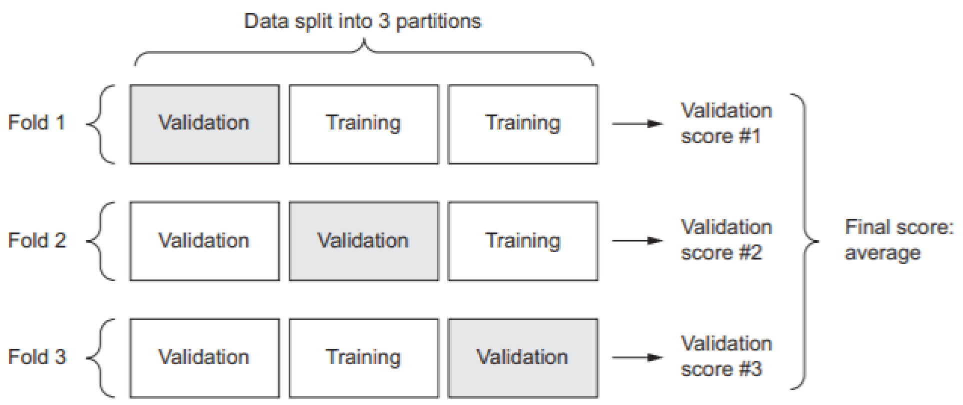


Figure 10: 3-fold cross-validation.

As is shown in picture, 3-fold cross-validation method splits the training dataset into three groups and implements modeling for three times. During each modeling, two groups are regarded as training data and the remaining one act as validation data. The validation score of the final model is the average value of these three results. This method is helpful especially when the there is a lack of training data.
However, as was told before, we have 323,478 pairs of question. What's more, I tried cross-validation method and the process was not efficient. As a result, training dataset is only splitted into training data and validation data with a rate of 8:2. But before splitted, the training dataset is shuffled to get a better model.

3.5 Model fitting and parameter tuning

As I mentioned above, there are several parameters that affect the quality of the model. As a result, they are supposed to be tuned to get the best model. There are four main parameters that I will tune in this section, the dropout rate of the Dropout layer, the output dimensionality of the LSTM layer, the number of hidden units of the Dense layer and the epochs of the neural network. It is desirable to use Gridsearch method to tune these four parameters simultaneously. However, it runs extremely slowly for my neural network and it is also helpful to tune these parameters one by one.

3.5.1 Tuning the dropout rate of the Dropout layer

At first, only the dropout rate is taken into account and it is tuned according to the accuracy and the loss of the model. At the same time, the output dimensionality of the LSTM layer, the number of hidden units of the Dense layer and the epochs of the neural network are fixed at 200, 125, 10 respectively.

dropout_rate	accuracy	loss
0.15	0.8271	0.3939
0.2	0.8261	0.4009
0.25	0.8159	0.3993
0.3	0.8088	0.4045
0.35	0.8074	0.4027
0.4	0.7920	0.4237

Table 1: Tuning the dropout rate of the Dropout layer.

As we can see from table 1, as the dropout rate increases, the overall accuracy of the model decreases and the loss increases. Therefore, 0.15 or 0.2 is the best value of the dropout rate.

3.5.2 Tuning the output dimensionality of the LSTM layer

Then the output dimensionality of the LSTM layer is tuned and the dropout rate of the Dropout layer, the number of hidden units of the Dense layer and the epochs of the neural network are fixed at 0.15, 125, 10 respectively.

num_lstm	accuracy	loss
175	0.8271	0.3956
185	0.8271	0.4147
195	0.8281	0.4054
205	0.8291	0.4014
215	0.8276	0.3996
225	0.8277	0.4024
235	0.8222	0.4198
245	0.8269	0.3994

Table 2: Tuning the output dimensionality of the LSTM layer.

As we can see from table 2, the accuracy and the loss of the model does not vary a lot as adjustment of the output dimensionality of the LSTM layer. Therefore, it is fixed at 205 to get the highest accuracy.

3.5.3 Tuning the number of hidden units of the Dense layer

Then the output the number of hidden units of the Dense layer is tuned and the dropout rate of the Dropout layer, dimensionality of the LSTM layer and the epochs of the neural network are fixed at 0.15, 205, 10 respectively.

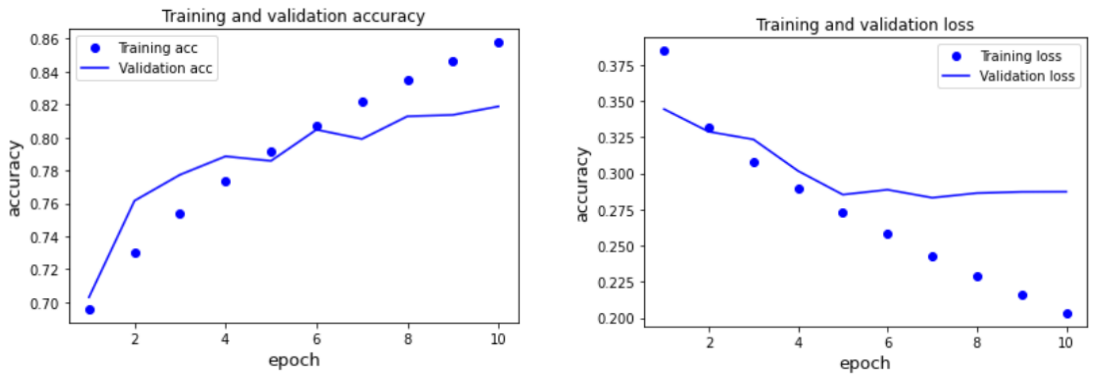
num_lstm	accuracy	loss
100	0.8159	0.4193
105	0.8204	0.4144
110	0.8277	0.4061
115	0.8237	0.3987
120	0.8297	0.3946
125	0.8291	0.4014
130	0.8260	0.4079
135	0.8297	0.4006

Table 3: Tuning the number of hidden units of the Dense layer.

As we can see from table 3, as adjustment of the number of hidden units of the Dense layer the accuracy and the loss of the model does not have a drastic fluctuation. When the number of hidden units is equal to 120, the model has the highest accuracy.

3.5.4 Tuning the epochs of the neural network

At last, the number of the epochs of the neural network also should be tuned. If it is set too low then it will cause underfitting. However, if it is set too high then it will result in overfitting. To tune it , the dropout rate of the Dropout layer, dimensionality of the LSTM layer and the number of hidden units of the Dense layer are fixed at 0.15, 205, 120 respectively. Figure 11 shows the change of the accuracy and the loss of the model with the number of epochs changing from 1 to 20.



(a) Accuracy (b) Loss

Figure 11: Accuracy and loss of the model of different epochs.

As we can see from these two plots, with the number of epochs increases, the accuracy of the validation data increases as well. However, when the the number of epochs reaches at around 10, the accuracy commences stabilizing. While for the loss of the model, at first it decreases as the number of epochs increases and then increases with the number of epochs keep going up, which resulting from overfitting. The best value of the number of epochs is 10, with which the model can have a high accuracy and avoid overfitting.

3.6 Evaluating the quality of the model

Now we have already tuned the parameters and as I mentioned above the output from 'sigmoid' dense layer is the probability that the inputted two questions are duplicate. However, we need come up an approach to use these probabilities to decide if question pairs are duplicate. The method that I use is picking a threshold, which is desirable and simple. Pairs whose probability is higher than threshold are classified as duplicate questions while pairs whose probability is less than threshold are declared as distinct questions.

In order to decide how to set the threshold, we need to know what kind of errors resulting from different threshold. If we set threshold too high, then there will be more false negative samples, that is, the inputted pair questions are duplicate but we declare them not. Conversely, if we set threshold too low, we will get more false positive samples, that is, the pair questions are distinct from each other but classified as duplicate falsely.
When we choose different threshold, the way I choose to evaluate the quality of the model is the common metrics precision and recall. Precision stands for the proportion of true duplicate question pairs among the pairs that are declared as duplicate questions, which can be computed as equation 3.5.

$$precision = \frac{TP}{TP + FP} \tag{3.5}$$

where TP is true positive and FP is false positive.
Recall represents the proportion of the question pairs that are declared as duplicated questions among all the true duplicated question pairs, which can be computed as equation 3.6.

$$recall = \frac{TP}{TP + FN} \tag{3.6}$$

where TP is true positive and FN is false negative.
Figure 12 shows the precision and recall at different level of threshold.

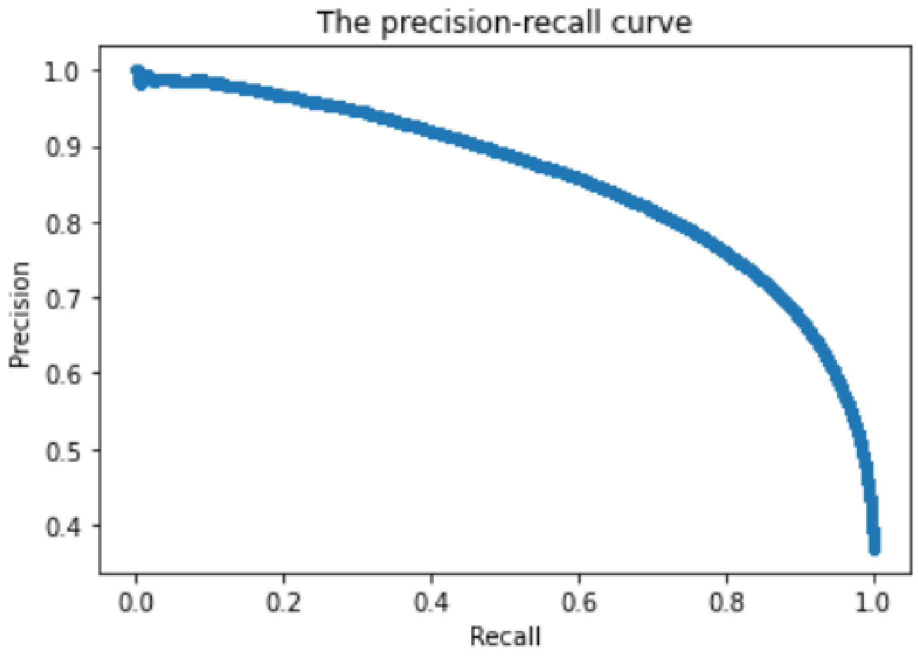


Figure 12: The precision-recall at different level of threshold.

There is a trade-off between precision and recall as we adjust threshold in that the false positives and false negatives are to be balanced. A third metric, named F-measure, takes the harmonic mean of precision and recall to evaluate overall quality of the model. F-measure can be computed as equation 3.7.

$$F - value = 2\frac{precision * recall}{precision + recall} \tag{3.7}$$

Figure 13 shows the precision, recall and the corresponding f-value at different level of threshold.

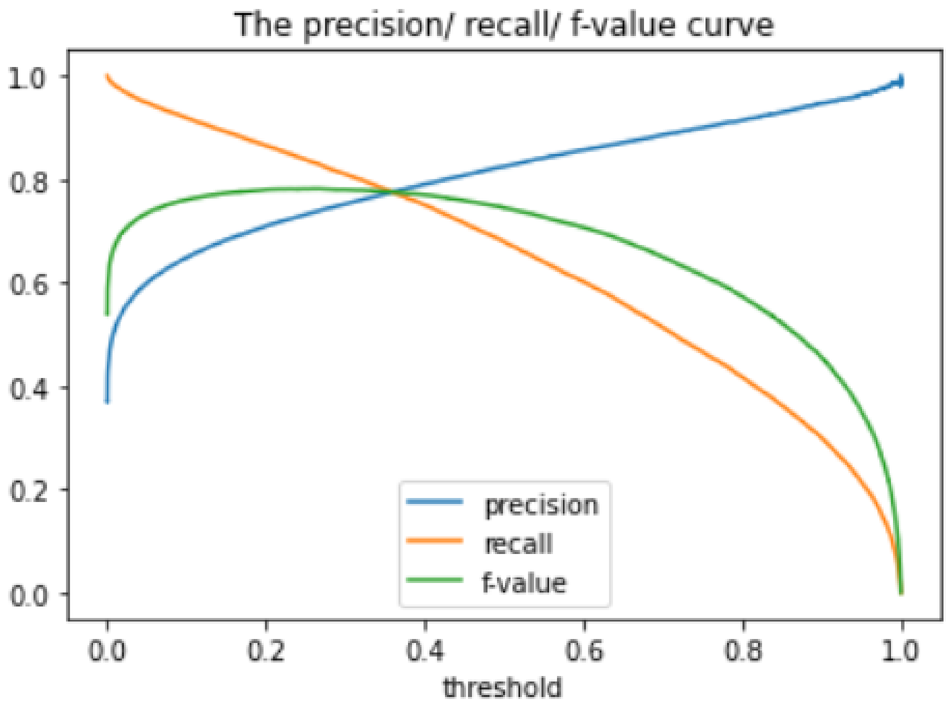


Figure 13: The precision, recall and f-value at different level of threshold.

As we can see from the plot, the maximum of f-value is approximately 0.8 when the threshold is around 0.38. In this case, both the value of precision and recall is around 0.8 as well. Figure 14 shows the confusion matrix in this case.

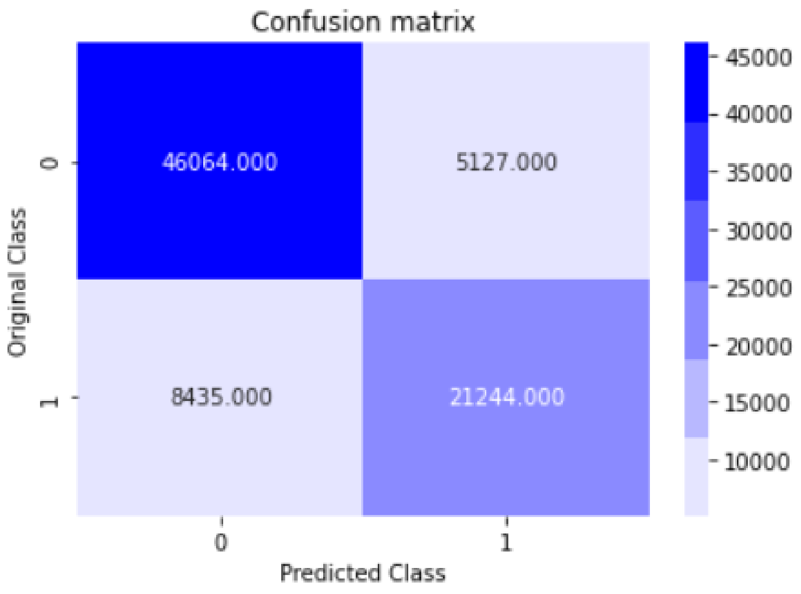


Figure 14: The confusion matrix when the threshold is equal to 0.38.

Another metric to evaluate the quality of the model is AUC (Area under the ROC curve). An ROC curve (receiver operating characteristic curve) shown the performance of the classification model at different thresholds. This curve plots False positive rate against True positive rate, these two parameters can be computed as equation 3.8.

$$\begin{aligned} TPR &= \frac{TP}{TP+FN} \\ FPR &= \frac{FP}{FP+TN} \end{aligned} \tag{3.8}$$

where TP is true positive, FN is false negative, FP is false positive and TN is true negative. As the increase of threshold, both true positive rate and false positive rate tend to decrease. AUC measures the area underneath the ROC curve and its value ranges from 0 to 1. What we want to find is a threshold at which false positive rate is low and true positive rate is high. As a result, AUC is expeted to be close to 1. Figure 15 shows the ROC curve.

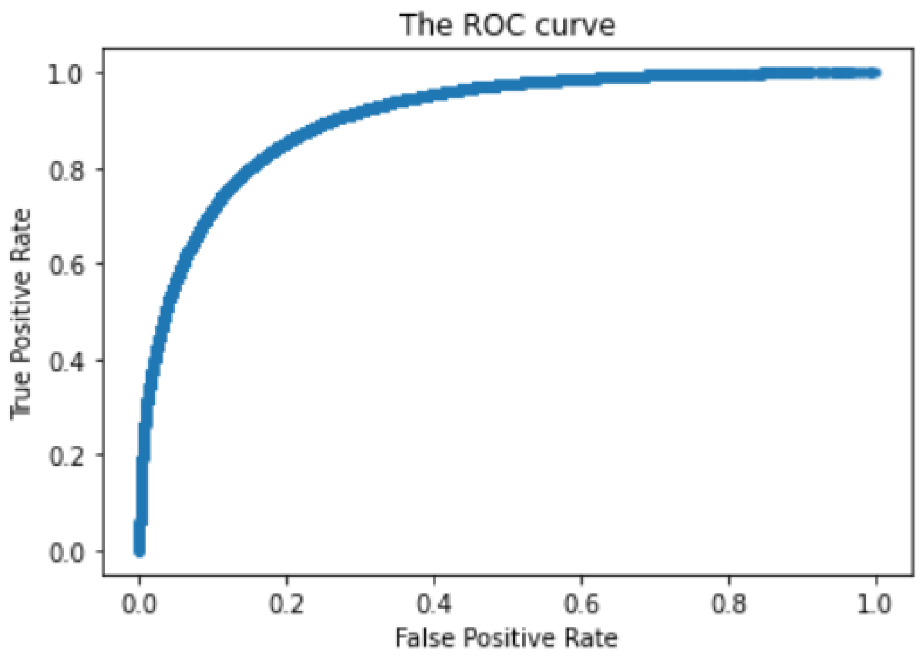


Figure 15: The receiver operating characteristic curve.

The AUC computed from this ROC is 0.910, which represents the quality of the model is desirable.

4 Conclusions

On some question answering websites, there are some duplicate questions. Under this circumstance, responders tend not to answer repeated questions, thus some users are less likely to get high-quality answers. This report uses siamese neural network to process the question pairs with the same parameters and weights. The key components of this network are pretrained word2vec embeddings and the LSTM layers in siamese neural network. After fitting the model, precision, recall and f-value are computed to evaluate the quality of the model. Besides, ROC is also plotted and AUC is computed. To our delight, the results are desirable and majority of duplicate questions are identified. Once the duplicate questions are predicted, it saves the energy and time of the responders on question answering websites and lead users to high-quality answers directly.

References

[1] Karl O'hare, K., Jurek-Loughrey, A. de Campos, C., 2019. An unsupervised blocking technique for more efficient record linkage. Data Knowledge Engineering, 122, pp.181–195.
[2] Sharma, L. et al., 2019. Natural Language Understanding with the Quora Question Pairs Dataset.
[3] Bromley, Jane, et al. "Signature Verification Using A "Siamese" Time Delay Neural Network."IJPRAI 7.4 (1993): 669-688.
[4] Wang, Shuohang, and Jing Jiang. "A Compare-Aggregate Model for Matching Text Sequences." arXiv preprint arXiv:1611.01747 (2016).
[5] "Understanding LSTM Networks" Colah's blog. Aug 27, 2015. Web. 24 Apr. 2017..

Appendices

A Python code

The code for this project is stored in the link below:
https://github.com/Allen-CongWang/Quora$_{q}uestion_{p}airs$