The School of Mathematics

THE UNIVERSITY *of* EDINBURGH

A Mahattan LSTM Based RNN Architecture for Paraphrase Identification

by

Yifan Zeng, s1998233

August 2020

Supervised by
Dr.Daniel Paulin, Dr.Aleksander Kolev and Grant Galloway

Acknowledgments

I am very grateful to the supervisors of this project: Dr.Daniel Paulin and Dr.Aleksander Kolev. I also wanna say thanks Grant Galloway, he provides the data and the background information of this subject.

Own Work Declaration

I confirm the work is done on my own and only supported by declared resources.
Yifan Zeng
Chengdu, China 6th August 2020

Contents

Executive summary

Quora[1] website is a platform for asking questions, and it connects answer contributors and questioners. Because of the enormous number of questions arising everyday, detecting duplicate questions with high precision can effectively save time for all users. Pairing same Quora questions together is a typical paraphrase identify question.

In this report, our goal is to build a model which can tell whether two questions are duplicate or not based on training with a 404K-size labelled data set. We discuss the original Manhattan LSTM model and a new Manhattan LSTM based RNN(Recurrent Neural Network) model, evaluate and compare their capacity with Keras computed `binary_accuracy`, the latter one with a much higher accuracy value(0.8829 vs 0.7902 on test set) is applied for this task.

This report starts by taking an overview of state-of-art approaches for this problem, and then makes exploratory analysis for training set. Subsequently derive three simple features along with the main feature, which is `Manhattan Distance` computed from original Manhattan LSTM model. And the three simple features are `Word Match Ratio`, `Q1 Q2 intersect`, and `TF-IDF weight`). They are processed to our Manhattan LSTM based RNN model. Finally we run experiments on this model and achieve an accuracy of 0.8829 on test set.

[1]https://www.quora.com/

1 Introduction

Paraphrase identification is the task of identifying whether two sentences are the same and it is a well-studied NLP task. In business context such as Quora website or Piazza forum, it's useful to improve users' experience. For example, the question you confronted with may have been asked and answered by other users before, then system could pop up similar question suggestions for you. Hence you can get the answer immediately and answer contributors don't need to answer same questions repeat. Furthermore, paraphrase identification technology is also used in many other fields such as machine translation, information retrieval and answer selection question Chandra & Stefanus (2020).

There are extensive research on this task. Some non deep learning method are popular in the past few decades. Bilotti et al. (2007) presents a structured retrieval techniques to retrieve more relevant results than bag-of-words retrieval, it works by encoding linguistic and semantic content as annotations on text. But this technique only benefit when sentences quantity is small and the prediction accuracy is relatively low. Then researchers pay more attention to exploiting semantic structure. Wang et al. (2007) propose a probabilistic quasi-synchronous grammar to augment existing classifier. Heilman & Smith (2010) use tree edit models to extract sequences and then describe a logistic regression model that uses 33 syntactic features of edit sequences to classify the sentence pairs. Instead of focusing on high-level syntactic presentation, Yao et al. (2013) turned attention to shallow level representation and extend word-to-word alignment to phrase-based aligner by a semi-Markov CRF. However, almost all non deep learning methods require large computation resources. Besides, classifier based on syntactic or semantic parsers to find the best match between structured presentations is not trivial Yin et al. (2016).

Then it comes to the deep learning, or the neural network-based era of paraphrase identification. Two types of neural-based frameworks are proposed in recent years. The first is based on a 'Siamese' network which is consisting of two sub-networks, these two sub-networks share same weight at each level and aim at extracting features from two input sentences. The classifier makes final decision based on these two vectors generated by two sub-networksChen et al. (2018). The second is based on 'Compare-aggregate' which considering the interactions between sentences. In terms of the fact that compare-aggregate models perform matching in single direction, Wang et al. (2007) uses a bilateral multi-perspective matching (BiMPM) model, matches the encoding in two directions and aggregate the pairing results using another BiLSTM layer. Asides 'Siamese' and 'Compare-aggregate' frameworks, another effective approach called 'attention-based' is also used handling this task, which models the interdependence between the two sentences in a sentence pairYin et al. (2016). One of the well-performing attention-based model is Attention Based Convolutional Neural Network (ABCNN) for modeling a pair of sentences. It captures information from sentences at different levels of abstractions by stacking multiple convolutional layers Yin et al. (2016).

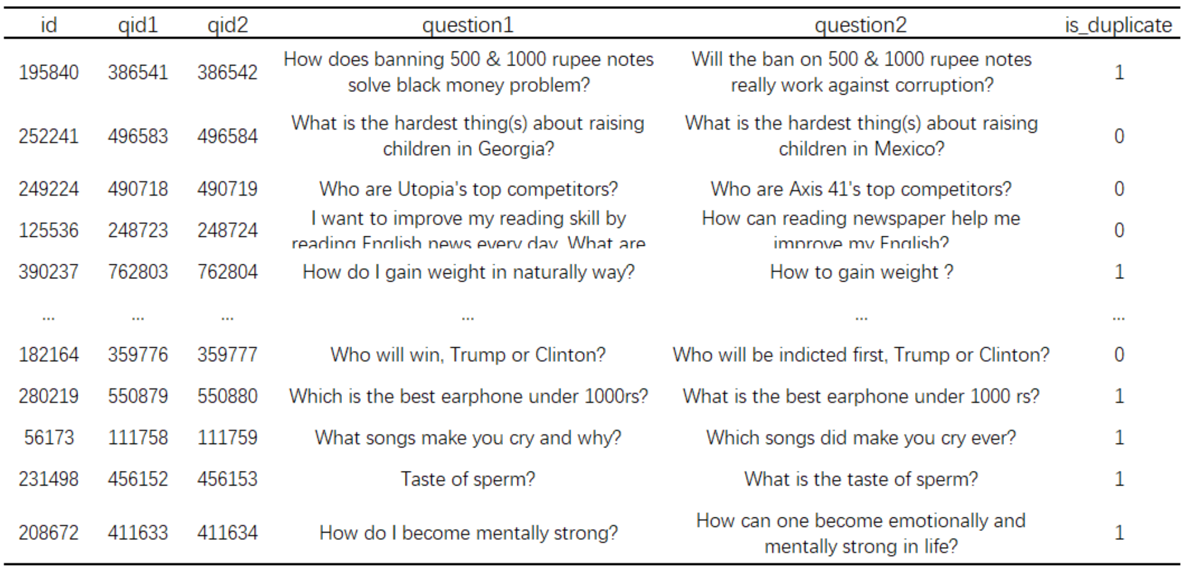
1.1 Related Work

Mueller & Thyagarajan (2016) presents an approach called MaLSTM which is short for Manhattan LSTM. It relies on pre-trained word-vectors for two LSTM inputs and restricts subsequent operations on a simple Manhattan metric. The sentence representations learned by MaLSTM form a highly structured space and the geometry distance reflects complex semantic relationship. Unlike typical language modeling RNNs, where LSTM are used to predict the next word given the previous text, here LSTMs simply function like the encoder. This model also demonstrates that a simple LSTM is capable of modeling complex semantics if the representations are extracted explicitly guided. Othman et al. (2019) applies MaLSTM on real world `Yahoo!` Answers dataset and shows the efficiency of the method both in Arabic and English languages.

Chen et al. (2018) proposes a siamese architecture based on MaLSTM, they add a classification step after computing Manhattan distance with LSTM representations. The classification step is made by 3-layer neural networks and this implementation does improve the model. Godbole et al. (2018) combines MaLSTM with classic machine learning classification methods such as Random Forest, Adaboost and SVM, however the improvement is not that much than Chen et al. (2018). It has to be mentioned that, both Chen et al. (2018) and Godbole et al. (2018) consider adding other features suggested by `Kaggle` forum as well as the output from MaLSTM.

2 Data Analysis

In this section, we take a general overview of the data set and introduce some of our observations of the data set. These observations help us have a more profound understanding of the data and also provide more idea in the later analysis process.



id	qid1	qid2	question1	question2	is_duplicate
195840	386541	386542	How does banning 500 & 1000 rupee notes solve black money problem?	Will the ban on 500 & 1000 rupee notes really work against corruption?	1
252241	496583	496584	What is the hardest thing(s) about raising children in Georgia?	What is the hardest thing(s) about raising children in Mexico?	0
249224	490718	490719	Who are Utopia's top competitors?	Who are Axis 41's top competitors?	0
125536	248723	248724	I want to improve my reading skill by reading English news every day. What are	How can reading newspaper help me improve my English?	0
390237	762803	762804	How do I gain weight in naturally way?	How to gain weight ?	1
...	...	...	...		...
182164	359776	359777	Who will win, Trump or Clinton?	Who will be indicted first, Trump or Clinton?	0
280219	550879	550880	Which is the best earphone under 1000rs?	What is the best earphone under 1000 rs?	1
56173	111758	111759	What songs make you cry and why?	Which songs did make you cry ever?	1
231498	456152	456153	Taste of sperm?	What is the taste of sperm?	1
208672	411633	411634	How do I become mentally strong?	How can one become emotionally and mentally strong in life?	1

Figure 1: Pairs in train set.

2.1 Overview

The whole data set consists of 404K question pairs, there are 80818 and 323259 sample pairs in the test set and training set respectively. Each question pair contains a unique pair ID, question_1 ID and question_2 ID as Figure1 showing. Two questions are presented with full text such as 300402: 'How do you get rid of a virus on an iPhone?' and 300403: 'How do I get rid of a virus on my iPhone?'. And for each pair, there is a label column called 'is_duplicate' showing whether these two questions are same or not. The definition of duplicate by Quora website is that these two questions can be solved by same answers.

In terms of the scope of questions, these sample questions almost cover all fields such as technology, religion and normal life etc. It's reasonable because of the enormous number of sample size and `Quora` website which has lots of users is really well-known. As a result there are questions contains special characters, mathematical signs, abbreviations or non-english words.

It should be mentioned that the labels for each pair is decided by human experts. So the real meaning of each question can never be known with full certainty which may induce bias.

2.2 Analysis

For training set, the number of duplicate pairs is 119551 which covers only 37% of all training sample pairs indicating the sample is unbalanced and there are more non-duplicated question pairs. In addition, 20% questions appear multiple times in different question pairs Chen et al. (2018). Below are some other more specific observations.

The first observation is that the sample pairs are noisy. Some question full text contains not only one question, take 439089 for example: 'Does the double helix structure of DNA serve a functional purpose? What if it were flat?', then it is not duplicate with just its sub-question. Besides, there are some typos in the text, human experts can infer from the typos and give right label. For example, 1613: 'How do I know if the girl am dating online truly live me for real?' and 1614: 'How do I know if my online girlfriend truly loves me?' are duplicate and apparently 'live' is typo for 'love'. But it may be uneasy for model to notice it. Also, some in-duplicate questions are almost the same but are different at just one key word, such as 72896: 'How can I study GRE in 2 month?' and 72897: 'How can I study GRE in 3 month?'.

The second observation is that the order of words is also an important factor for labelling. There is a good example of 'Why is Google Chrome not working but Internet Explorer is' and '"Why is Internet Explorer not working but Google Chrome is' Chen et al. (2018). These two questions have same word list but different order then totally different meaning. It tells us simple word-to-word match can't work perfectly for this task.

The third observation is that almost all question pairs share the same start word with interrogative words such as Why, How, What and Can etc. and end with question mark. This type of similarity are not significant feature to attach label because there's no huge difference. We can consider removing useless part in texts.

3 Feature Engineering

The idea of solving paraphrase identification problem is to detect sentences semantic similarity and set threshold to attach labels. The main feature in our model used for classifying is extracted by Manhattan LSTM part, aside from it three simple features `Word Match Ratio`, `Q1 Q2 intersect`, and `TF-IDF weight` are also considered into our secondary classifier. These three features are not only suggested by users on`Kaggle` forum, but also have been included in previous researches. Our framework can also benefit from adding these features after experiments. Below is more information about these three features.

`Q1 Q2 Intersect`: Although classifying only by word-to-word match can't perfectly handle the question pairs, common words still appear frequently in duplicate question pairs. So the length of largest continuous word set which contains all the common elements in both Q1 and Q2 is considered as a useful feature for semantic similarity detection.

`Word Match Ratio`: The Q1 Q2 Intersection only tells how many common words there are between questions, it doesn't take the total length of question itself into consideration. So we add a ratio feature to reflect this feature. Word Match Ratio is the shared words number divided by the total number of question texts after removing stop-words.

`TF-IDF Weight`: Sometimes shared words between questions can't give a significant influence on semantic meaning such as 'Why', 'What' or 'Can' etc. So we consider distinguish relatively more useful common words from meaningless shared words by weighting each word, a technology TF-IDF would help here. For all words from question text in training set and test set, Term Frequency-Inverse Document Frequency metric(TF-IDF) weights the common words in question texts and gives scores to each word. It calculates the addition of common word weights over total weight.

4 RNN-LSTM

Neural Networks are computing systems that vaguely inspired by biological neural network that constitute animal brainChen et al. (2019). This kind of system is consisting of nodes which work similarly as neurons. Each node process signals entries with non-linear functions, and then pass the signal output to the next node via edges between nodes. Several nodes form layers, usually layers play different roles in networks such as input layers, hidden layers, dropout layers or output layers. The weight between edges is important for neural networks and it is exactly what we are training with data.

Recurrent Neural Network generalise feed-forward neural network by adding internel memory which make it capable of processing unsegmented and connected sequences. For RNN all inputs are not independent, feedback connections enable RNN to propagate data from earlier events to current processing steps, which means it can connect previous outputs with new inputs Staudemeyer & Morris (2019). Given this advantage, RNN is considered to handle sentence similarity task in terms of relationship inside sentences. However, RNN can't handle very long sequences because of gradient vanishing and exploding problems, and decreases in performance with sentence length increasing.

For this restriction, Long-Short Term Memory(LSTM) structure appears as a modified version of RNN. LSTM adds a forget gate to discover what detail to be discarded. So it offers more flexibility to decide what information to store in memory and is more suitable for longer sequences such as questions in this task.

4.1 Siamese Neural Network

Siamese network is an architecture that is consisting of two sub-networks. The notable feature for Siamese network is that these two sub-networks share same weight. It is widely used for tasks that aim at detecting similarity between two inputs. The idea of Siamese networks is that if two inputs are duplicate or are of high similarity, their representation vectors will be very close. Hence there should be a distance function to valuate the similarity. And the goal of training this model is to make duplicate inputs as close as possible and make in-duplicate inputs far away with each other. In this report we consider using the power of Siamese network to detect sentences similarity.

4.2 Manhattan LSTM

As shown in Figure2 Manhattan LSTM is an adaption of both Siamese Neural Network and LSTM as Siamese networks are good at detecting similarity and LSTM are useful for long sequences. Two LSTM networks of same weights are used to tackle two inputs, and the inputs can be two question texts in our task, they map variable-length question text into fixed-length vector representation. Then a pre-defined similarity function is applied to the outputs of LSTM networks, this function gives a value to reflect the semantic similarity of inputs. The name "Manhattan" here stems from the function which applies the manhattan distance metric Mueller & Thyagarajan (2016).

4.3 Dropout Layer

In machine learning fields. Regularization such as Lasso regression and Ridge regression method are widely used to avoid capturing too much random information from training data hence prevent overfitting. Overfitting is also common for this task, dropout layers between hidden

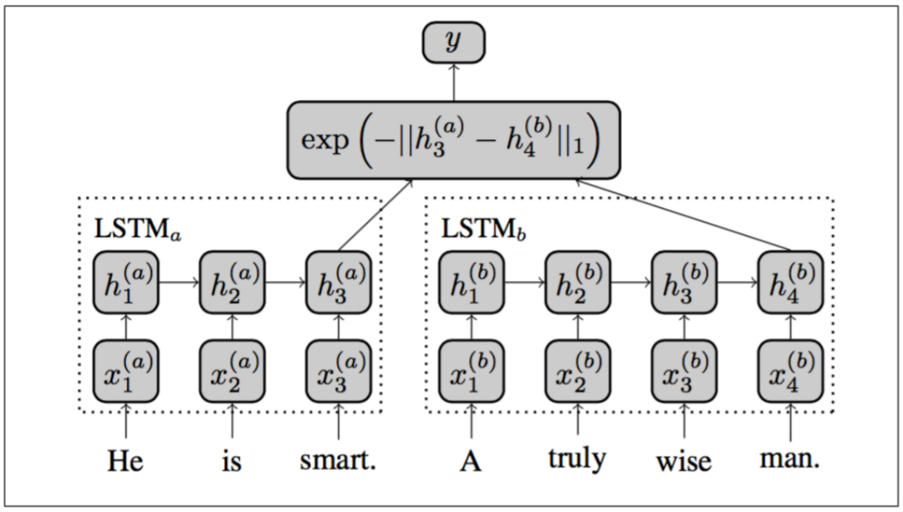

y
$\exp\left(-\|h_3^{(a)} - h_4^{(b)}\|_1\right)$
LSTM$_a$
LSTM$_b$
$h_1^{(a)}$ $h_2^{(a)}$ $h_3^{(a)}$
$x_1^{(a)}$ $x_2^{(a)}$ $x_3^{(a)}$
$h_1^{(b)}$ $h_2^{(b)}$ $h_3^{(b)}$ $h_4^{(b)}$
$x_1^{(b)}$ $x_2^{(b)}$ $x_3^{(b)}$ $x_4^{(b)}$
He is smart.
A truly wise man.

Figure 2: Overview of Manhattan LSTM framework.Mueller & Thyagarajan (2016)

layers in neural networks are useful to handle this problem. The main idea of dropout is adding noise in output value by randomly deleting part of nodes temporarily. Dropout layer work by setting some elements of output vectors as zero in training process which means dropping the information extracted by these nodes. With a dropout rate which is usually between 0.2 to 0.5, the proportion of nodes to be set as zero is fixed.

4.4 My Model Framework

Inspired by the Manhattan LSTM, we build our model. It's a combination of Manhattan LSTM and RNN model. As Figure3 shown, the **first part** of our architecture also consists of two Siamese LSTM sub-networks (one for each question in question pairs) and uses Manhattan distance metric as similarity function. Then a concantenate layer is used to combine output of Manhattan LSTM, feature `Word Match Ratio`, feature `Q1 Q2 Intersect` and feature `TF-IDF Weight`. Subsequently these four extracted information is fed into the **second part**, which is a 3-layer neural network(with 2 dropout layers) to make the final decision of duplicate or not, by this way our architecture considers more information from question texts than traditional Manhattan LSTM which just uses Manhattan distance.

The structure starts with two input layers, each layer pass the representation vector of question into the sequence embedding layer. Then each embedding layer generates sentence embedding vector and pass two vectors into LSTM network. As we are using a Siamese based network, these two LSTM networks surely have same weights. Subsequently a single vector representation is obtained by Manhattan similarity function and is computed as part of input for the secondary classifier.

The concantenate layer is used to connect output from Manhattan LSTM and the three simple features mentioned above, then all these features are ready for classification part. The classifier is conducted using a 3-layer neural network consisting of an input layer, a hidden layer and a output layer. Dropout layers are considered in our model to avoid serious overfitting. Finally the output layer gives a probability of similarity for two input questions.

5 Experiments and Results

5.1 Pre-process

Upon loading the train set into our model, we start with pre-processing the text. Firstly we remove specific signs and extend some abbreviations into full expression. Then we lower each

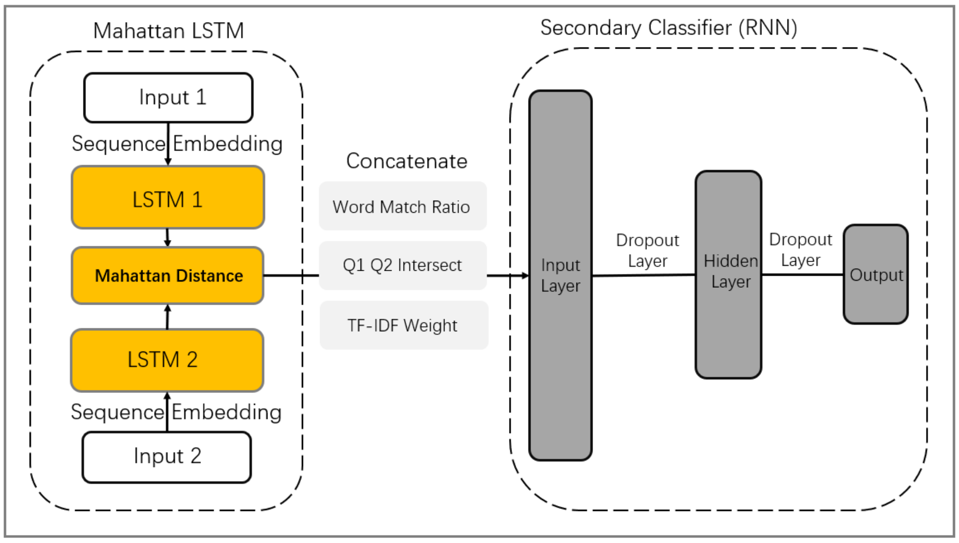


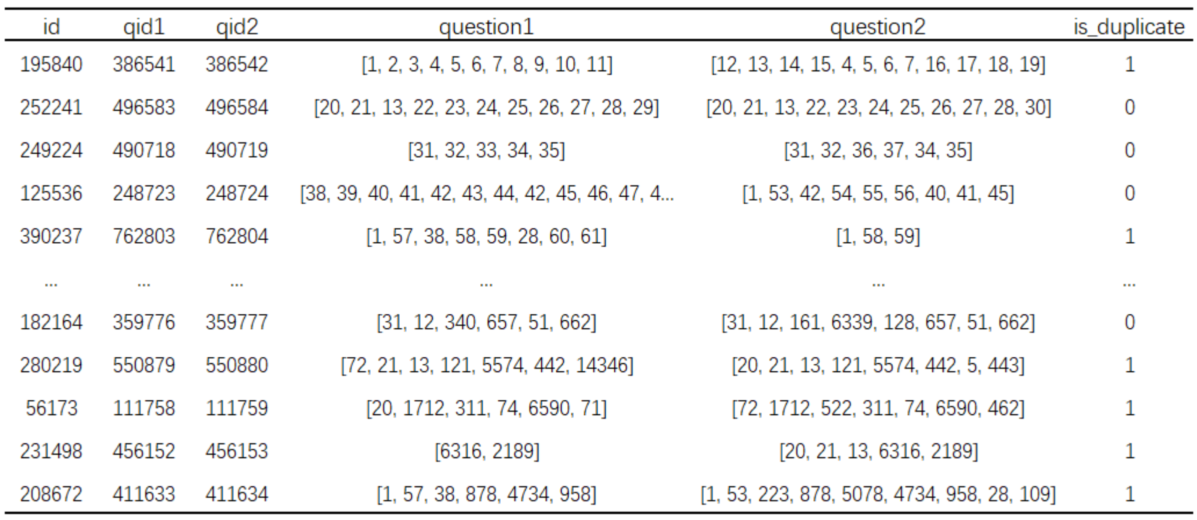
Figure 3: Overview of my model framework.

word in question text and split the text string into list with words as elements.

```python
def question_to_word(text):
    ''' Lower the capitals
        Preprocess the abbreviation
        Convert question text to a list of words '''
    text = str(text)
    text = text.lower()
    # Clean the text (some similar rows are omitted below)
    text = re.sub(r"[^A-Za-z0-9^,!.\/'+-=]", " ", text)
    text = re.sub(r"what's", "what is ", text)
    text = re.sub(r"\'s", " ", text)
    text = re.sub(r"\'ve", " have ", text)
    text = re.sub(r"can't", "cannot ", text)
    ...
    ...
    text = text.split()

    return text
```

As the questions text are symbols which is English language, but the model can only processes in value expression. So before embedding, the question text need to be represented with word indices which are unique ids(integer) for different words. We created a dictionary object which takes word(string) as keys and indices(integer) as values for words in texts except stop words which are not included by embedding metric. It has to be mentioned that indice 0 or word id 0 is not aligned to any word because it represents unknown word and will be useful in subsequent zero padding step. Then the data set is shown as Figure.4. The final step before embedding is zero padding which extend the length of each question list with zeros to make question lists of same length as required by embedding. The max-length of questions among all sets is 212, so we zero padding each question indices list to 212 dimension.
Before training, we compute the value for other three simple features. Now five extraction consists of two presentation vectors of questions and three simple features are prepared for model experiment.

```python
#code for feature Q1 Q2 intersect:
len(question1.intersection(question2))
#code for feature Word Match Ratio:
len(question1.shared.words)+len(question2.shared.words)/len(question1+question2)
#code for feature TF-IDF Weight:
sum(shared.words.weights)/sum(all.words.weights)
```



id	qid1	qid2	question1	question2	is_duplicate
195840	386541	386542	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]	[12, 13, 14, 15, 4, 5, 6, 7, 16, 17, 18, 19]	1
252241	496583	496584	[20, 21, 13, 22, 23, 24, 25, 26, 27, 28, 29]	[20, 21, 13, 22, 23, 24, 25, 26, 27, 28, 30]	0
249224	490718	490719	[31, 32, 33, 34, 35]	[31, 32, 36, 37, 34, 35]	0
125536	248723	248724	[38, 39, 40, 41, 42, 43, 44, 42, 45, 46, 47, 4...	[1, 53, 42, 54, 55, 56, 40, 41, 45]	0
390237	762803	762804	[1, 57, 38, 58, 59, 28, 60, 61]	[1, 58, 59]	1
...	...	...	...	...	...
182164	359776	359777	[31, 12, 340, 657, 51, 662]	[31, 12, 161, 6339, 128, 657, 51, 662]	0
280219	550879	550880	[72, 21, 13, 121, 5574, 442, 14346]	[20, 21, 13, 121, 5574, 442, 5, 443]	1
56173	111758	111759	[20, 1712, 311, 74, 6590, 71]	[72, 1712, 522, 311, 74, 6590, 462]	1
231498	456152	456153	[6316, 2189]	[20, 21, 13, 6316, 2189]	1
208672	411633	411634	[1, 57, 38, 878, 4734, 958]	[1, 53, 223, 878, 5078, 4734, 958, 28, 109]	1

Figure 4: The dataset before embedding.

5.2 Model Compile

To compile our structure, we start with input layers. As this is a multi-input model, Keras functional API is very useful to build it.

```
#input layers
input_q1 = Input(shape = (max_length,), dtype = 'int32')
input_q2 = Input(shape = (max_length,), dtype = 'int32')
input_wordmatch = Input(shape = (1,), dtype = 'float32')
input_intersect = Input(shape = (1,), dtype = 'float32')
input_tfidf = Input(shape = (1,), dtype = 'float32')
```

Now for embedding step, Word2Vec data is used for each question. Word2Vec is a pre-trained model on enormous amount of dictionaries and datasets, it's widely used to mapping words into high-dimension vectors and keeping the semantic similarity between words at the same time. Embedding matrix is defined with Word2Vec embedding values. The embedding dimension value is 300 in our structure which is same as Mueller & Thyagarajan (2016). Because we want to take use of Word2Vec mapping rule, 'trainable' argument should be False here.

```
#embedding layers
embedding_layer = Embedding(len(embed_matrix), embedding_dim = 300,
                            weights = [embed_matrix],
                            input_length = max_length, trainable = False)
embedded_q1 = embedding_layer(input_q1)
embedded_q2 = embedding_layer(input_q2)
```

Now it comes to LSTM part, the dimensionality value of the output space is 50 and it's also same as Mueller & Thyagarajan (2016). A function called 'manhattan_distance' is used for calculating, it defines by:

$$exp(-\|h_3^{(q1)} - h_4^{(q2)}\|_1)$$

```
#LSTM
shared_lstm = LSTM(50)
output_q1 = shared_lstm(embedded_q1)
output_q2 = shared_lstm(embedded_q2)

#function to calculate manhattan distance
def manhattan_dist(q1, q2):
    distance = K.exp(-K.sum(K.abs(q1 - q2),axis = 1, keepdims = True))
    return distance

manhattan_lstm_distance = Lambda(function = lambda x:manhattan_dist(x[0],x[1]),
                output_shape = lambda x:(x[0][0], 1))([output_q1,output_q2])
```

5.3 Model Training and Fine Tuning

5.3.1 Base model: Manhattan LSTM

We take the original version of Manhattan LSTM introduced by Mueller & Thyagarajan (2016) as base model for this task. Then the model is compiled as code below in which the classification decision is made only by manhattan distance value. The validation accuracy for this model is around 0.82 at most which is relatively lower than that of accuracy reached by models below. We take 0.82 value as the baseline accuracy of this task.

```
model = Model([input_q1, input_q2], [manhattan_lstm_distance])
model.compile(loss='binary_crossentropy',optimizer='rmsprop',metrics=['
    binary_accuracy'])
```

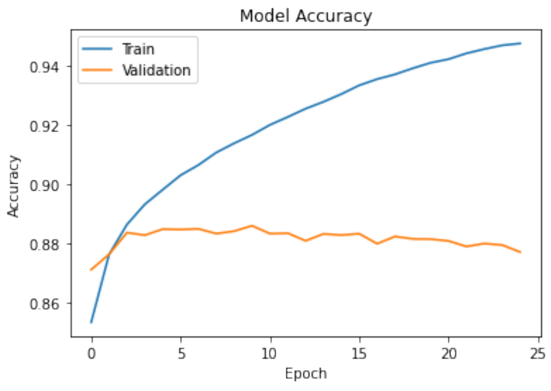
5.3.2 Updated model: Manhattan LSTM Based RNN

For Manhattan LSTM based RNN model, we concatenate the output from Mahattan LSTM part with other three simple features as a new input of the secondary classifier part. The activation function of the third layer nerual network is 'sigmoid' and use 'binary_crossentropy' and 'binary_accuracy' as loss function and fitting standard respectively that's because our task is a binary classify problem. The optimizer 'rmsprop' is suitable for almost all kinds of problems Chollet (2018).

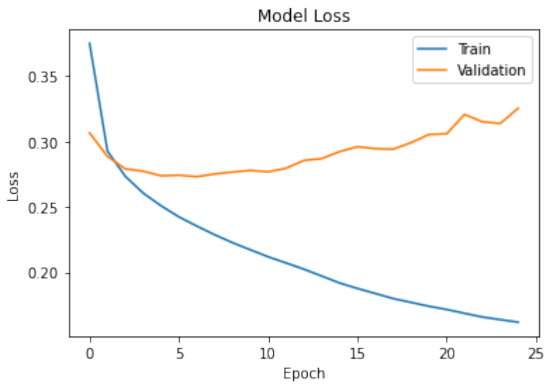
```
#concatenate layer
concatenated = layers.concatenate([manhatten_lstm_distance,
                    input_wordmatch, input_intersect,input_tfidf], axis = -1)

#3-layer neural networks
input_1 = layers.Dense(6, activation = 'relu')(concatenated)
hidden_1 = layers.Dense(3, activation = 'relu')(input_1)
output = layers.Dense(1, activation = 'sigmoid')(hidden_1)

#compile the model
model = Model([input_q1, input_q2,input_wordmatch,input_intersect,input_tfidf],[
    output])

model.compile(loss='binary_crossentropy',optimizer='rmsprop',metrics=['
    binary_accuracy'])
```

While training the model, we take validation split rate as 0.1. It means that 10% of training set are selected randomly as validation set continuously, in each epoch the trained model computes a validation accuracy on validation set. As model is unfamiliar with validation set, it's a good way to supervise overfitting without using information from test set. We look at accuracy computed by Keras default to observe if the model is overfitting

For the first time we train the model 25 epochs, then draw training and validation accuracy/loss plots. As we can see from Figure5, the training accuracy keeps increasing and almost reaches 1.0 at last while the validation accuracy is around 0.88 and stops increasing before epoch 10. Hence, we consider adding dropout layers between neural network layers to avoid overfitting and train the model for the second time.



(a) Training and Validation Accuracy



(b) Training and Validation Loss

Figure 5

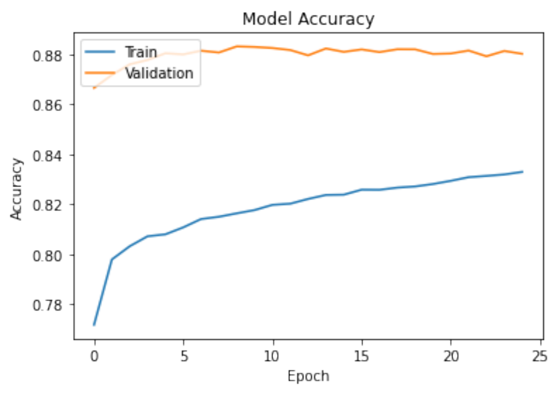
5.3.3 Updated model: Manhattan LSTM Based RNN with Dropout Layers

The model added two dropout layers with dropout rate 0.5 is compiled as:

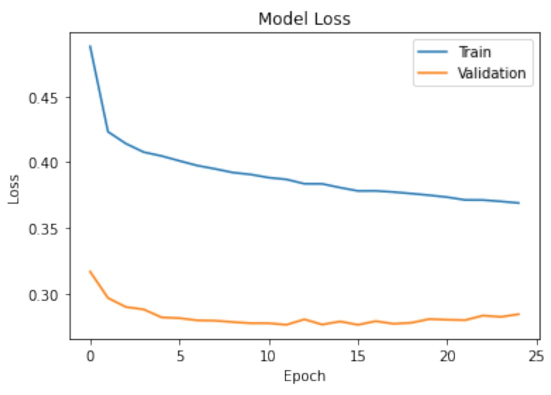
```
#concatenate layer
concatenated = layers.concatenate([manhatten_lstm_distance,
                    input_wordmatch, input_intersect,input_tfidf], axis = -1)

#3-layer neural networks with two dropout layers
input_1 = layers.Dense(8, activation = 'relu')(concatenated)
drop_out_1 = layers.Dropout(0.5)(input_1)
hidden_1 = layers.Dense(4, activation = 'relu')(drop_out_1)
drop_out_2 = layers.Dropout(0.5)(hidden_1)
output = layers.Dense(1, activation = 'sigmoid')(drop_out_2)

#compile the model
model = Model([input_q1, input_q2,input_wordmatch,input_intersect,input_tfidf],[
    output])

model.compile(loss='binary_crossentropy',optimizer='rmsprop',metrics=['
    binary_accuracy'])
```

Now we train the model with training set again, the validation rate and epoch times keep unchanged. The training and validation accuracy/loss curves are shown as Figure6. We can see that the validation accuracy is still around 0.88 while the training accuracy doesn't keep increasing with epoch times anymore. Instead, after 25 times epochs, the training accuracy is still 0.82 or so which means overfitting has been avoided. Now we use this model for test data set to check model capacity. It takes 1 hour to run 25 epochs.

At last, we are trying to improve our model by searching hyper parameters such as the dropout rate, the node number in the 3-layer networks and the LSTM model node number. However, the model performance haven't improve significantly so we keep these parameters unchanged as code blocks above showing.



(a) Training and Validation Accuracy



(b) Training and Validation Loss

Figure 6

5.4 Model Evaluation

For binary classification task, we always use value accuracy, precision, recall(sensitivity) and f1 score on test data set to evaluate our model. Before listing the corresponding values we make a brief introduction to them. There are some useful terms for computing these values. TP means true positive, it's the correctly predicted positive values in the model which means the value of predicted class and actual class are both positive. TN(True Negative), FP(False Positive) and FN(False Negative) are defined similarly.

Accuracy is the most intuitive measure and it's definition is simply a ratio of all correctly prediction observations to all observations. It's power when the data is not imbalanced.
Accuracy = TP + TN / TP + FP + TN + FN

Precision is the ratio of all correctly prediction observations to the total predicted positive observations. We can take it as that in all predicted duplicate question pairs, how many pairs are really duplicate. Precision = TP / TP + FP

Recall(Sensitivity) is the ratio of all correctly predition observations to the observations that are actual positive. It answers the question that in all duplicate question pairs how many are correctly labelled out by our model. Recall = TP / TP + FN

F1 score is a weighted combination of Recall and Precision. It's more useful especially when the data is not balanced. F1 score = 2*(Recall * Precision) / (Recall + Precision)

For our model, we got Accuracy at 0.882912, Precision at 0.904848, Recall at 0.760976 and F1 score at 0.826699. These four measures are all much higher than 0.5 which indicate a good performance of classification.

Besides, we use some plot to show the performance of our model. Roc curve tells us how True Positive Rate(TPR = TP / TP + FN) and False Positive Rate(FPR = FP / FP + FN) changing as threshold varying. The curve that is close to the top-left indicates good model so the area under curve is usually considered as a measure. Recall curve is depicted precision against recall which pay more attention to the positive observation label capacity of the model. The precision recall curve and roc curve are depicted in Figure7. It's easy to see that they are both smooth which indicate no serious overfitting. The area under roc curve and recall curve are both large, which are 0.938986 and 0.906986 respectively.

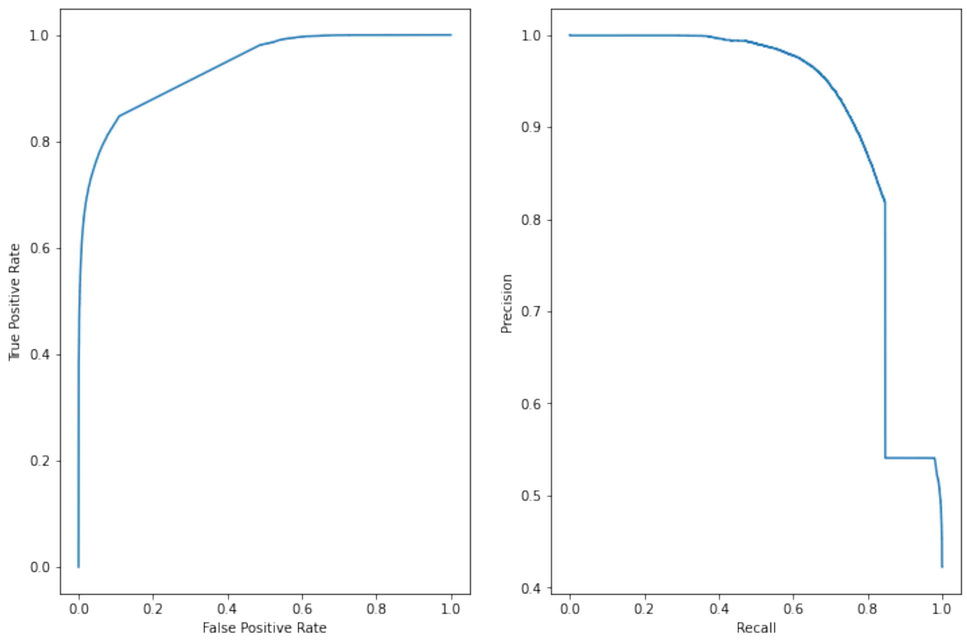


Figure 7: The precision recall curve.

6 Conclusion

In this report, our task is to build a model which can tell question pairs from Quora forum is duplicate or not. The data set contains 404K pairs with labels attached by human experts. This task is a paraphrase identification NLP problem, we solve it by detecting semantic similarity.

At the beginning of the report, we take an overview of previous researches in this field and make brief introduction of non deep learning methods and also deep learning approaches in the previous several decades. Besides, we talk about related work especially Manhattan LSTM and Siamese networks that inspire us. Then we introduce the data set and find some observations which may help in later step. Three simple features that help us complete our model are extracted, they are question intersection length, shared word ratio and TF-IDF weights. The framework of our model is a combination of two part, the first part is a Siamese Manhattan LSTM which take question text as input, after embedding and LSTM process it computes Manhattan distance between input representation vectors. The second part is a classifier consists of 3-layer Neural Network with 2 dropout layers which take the output from Manhattan LSTM as input along with other 3 simple features. The model is compiled by Keras function API and trained on Kaggle GPU.

The performances of the model on validation set and test set are both decent. We get accuracy at 0.882912 and f1 score at 0.826699. Further effort is required to explore other state-of-art methods such as BiMPM or ABFNN model.

References

Bilotti, M. W., Ogilvie, P., Callan, J. & Nyberg, E. (2007), Structured retrieval for question answering, *in* 'Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval', pp. 351–358.

Chandra, A. & Stefanus, R. (2020), 'Experiments on paraphrase identification using quora question pairs dataset', *arXiv preprint arXiv:2006.02648* .

Chen, Y.-Y., Lin, Y.-H., Kung, C.-C., Chung, M.-H., Yen, I. et al. (2019), 'Design and implementation of cloud analytics-assisted smart power meters considering advanced artificial intelligence as edge analytics in demand-side management for smart homes', *Sensors* **19**(9), 2047.

Chen, Z., Zhang, H., Zhang, X. & Zhao, L. (2018), 'Quora question pairs'.

Chollet, F. (2018), *Deep Learning mit Python und Keras: Das Praxis-Handbuch vom Entwickler der Keras-Bibliothek*, MITP-Verlags GmbH & Co. KG.

Godbole, A., Dalmia, A. & Sahu, S. K. (2018), 'Siamese neural networks with random forest for detecting duplicate question pairs', *arXiv preprint arXiv:1801.07288* .

Heilman, M. & Smith, N. A. (2010), Tree edit models for recognizing textual entailments, paraphrases, and answers to questions, *in* 'Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics', pp. 1011–1019.

Mueller, J. & Thyagarajan, A. (2016), Siamese recurrent architectures for learning sentence similarity, *in* 'thirtieth AAAI conference on artificial intelligence'.

Othman, N., Faïz, R. & Smaïli, K. (2019), Manhattan siamese lstm for question retrieval in community question answering, *in* 'OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"', Springer, pp. 661–677.

Staudemeyer, R. C. & Morris, E. R. (2019), 'Understanding lstm–a tutorial into long short-term memory recurrent neural networks', *arXiv preprint arXiv:1909.09586* .

Wang, M., Smith, N. A. & Mitamura, T. (2007), What is the jeopardy model? a quasi-synchronous grammar for qa, *in* 'Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)', pp. 22–32.

Yao, X., Van Durme, B., Callison-Burch, C. & Clark, P. (2013), Semi-markov phrase-based monolingual alignment, *in* 'Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing', pp. 590–600.

Yin, W., Schütze, H., Xiang, B. & Zhou, B. (2016), 'Abcnn: Attention-based convolutional neural network for modeling sentence pairs', *Transactions of the Association for Computational Linguistics* **4**, 259–272.

Appendices

A An Appendix

Code available under: https://github.com/YifanZeng1874/Dissertation_2.git