Quora Similar Question Detection

An Neural Network Approach

Georgetown University

MS Analytics Candidate
Cheng Zhong

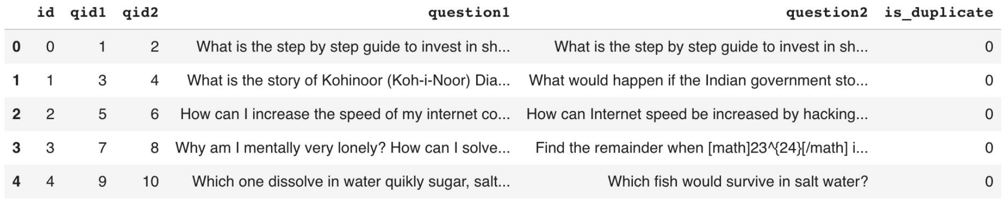
GEORGETOWN UNIVERSITY

Introduction

- Quora is a platform to ask questions and connect with people who contribute unique insights and quality answers
- Exists lots of people ask similarity worded questions which will significantly affect the user experience in finding the best answer

Goal & Purpose

- This project mainly focus on predicting whether a provided pair of questions has the same meaning using deep learning and NLP techniques



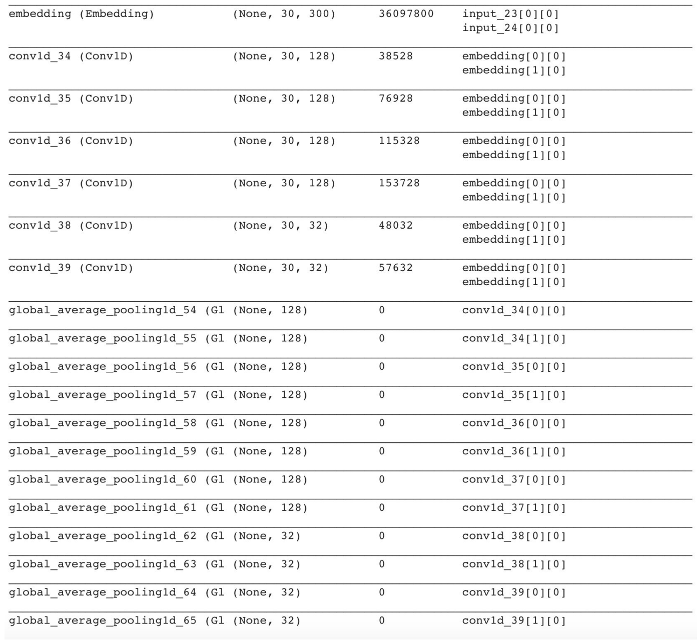
	id	qid1	qid2	question1	question2	is_duplicate
0	0	1	2	What is the step by step guide to invest in sh...	What is the step by step guide to invest in sh...	0
1	1	3	4	What is the story of Kohinoor (Koh-i-Noor) Dia...	What would happen if the Indian government sto...	0
2	2	5	6	How can I increase the speed of my internet co...	How can Internet speed be increased by hacking...	0
3	3	7	8	Why am I mentally very lonely? How can I solve...	Find the remainder when [math]23^{24}[/math] i...	0
4	4	9	10	Which one dissolve in water quikly sugar, salt...	Which fish would survive in salt water?	0

Figure 1. Training Data Example

Methodology

This project will apply several machine learning and deep learning method to classify the data and will use the log loss to evaluate the result. Since log-loss function heavily penalized classifiers that are confident about and incorrect classification. The loss score from Kaggle will be evaluate the final result.

- **Random Forest**
  - The classification method Quora is currently using
  - Simple, set as baseline of the evaluation
- **Multilayer Perceptrons (MLP)**
  - Easy to apply
  - Use the MLPClassifier from sklearn package
- **Convolutional Neural Network (CNN)**
  - Four convolutional layers
  - Four Global Average layers
  - Final Dense layer with size 512
  - Dropout rate 0.2



```
embedding (Embedding)            (None, 30, 300)      36097800    input_23[0][0]
                                                                  input_24[0][0]

conv1d_34 (Conv1D)               (None, 30, 128)      38528       embedding[0][0]
                                                                  embedding[1][0]

conv1d_35 (Conv1D)               (None, 30, 128)      76928       embedding[0][0]
                                                                  embedding[1][0]

conv1d_36 (Conv1D)               (None, 30, 128)      115328      embedding[0][0]
                                                                  embedding[1][0]

conv1d_37 (Conv1D)               (None, 30, 128)      153728      embedding[0][0]
                                                                  embedding[1][0]

conv1d_38 (Conv1D)               (None, 30, 32)       48032       embedding[0][0]
                                                                  embedding[1][0]

conv1d_39 (Conv1D)               (None, 30, 32)       57632       embedding[0][0]
                                                                  embedding[1][0]

global_average_pooling1d_54 (Gl (None, 128)           0           conv1d_34[0][0]

global_average_pooling1d_55 (Gl (None, 128)           0           conv1d_34[1][0]

global_average_pooling1d_56 (Gl (None, 128)           0           conv1d_35[0][0]

global_average_pooling1d_57 (Gl (None, 128)           0           conv1d_35[1][0]

global_average_pooling1d_58 (Gl (None, 128)           0           conv1d_36[0][0]

global_average_pooling1d_59 (Gl (None, 128)           0           conv1d_36[1][0]

global_average_pooling1d_60 (Gl (None, 128)           0           conv1d_37[0][0]

global_average_pooling1d_61 (Gl (None, 128)           0           conv1d_37[1][0]

global_average_pooling1d_62 (Gl (None, 32)            0           conv1d_38[0][0]

global_average_pooling1d_63 (Gl (None, 32)            0           conv1d_38[1][0]

global_average_pooling1d_64 (Gl (None, 32)            0           conv1d_39[0][0]

global_average_pooling1d_65 (Gl (None, 32)            0           conv1d_39[1][0]
```

Figure 2. CNN summary

Results

- **Exploratory Analysis:**
  - The total duplicated question pairs percentage is **36.92%** and the total useable question pairs for training is **404290**.
- **Random Forest:**
  - The Log-loss shown on the Kaggle public score board is **0.46317** on test data
- **Multilayer Perceptrons (MLP)**
  - The Log-loss shown on the Kaggle public score board is **0.47682** on test data
  - Similar performance as random forest
- **Convolutional Neural Network (CNN)**
  - The loss on training data is **0.1886** after 5 epochs
  - The Log-loss shown on the Kaggle public score board is **0.38594**
  - Better performance than the other two methods

```
Total number of question pairs for training: 404290
Duplicate pairs: 36.92%
Total number of questions in the training data: 537933
Number of questions that appear multiple times: 111780
```

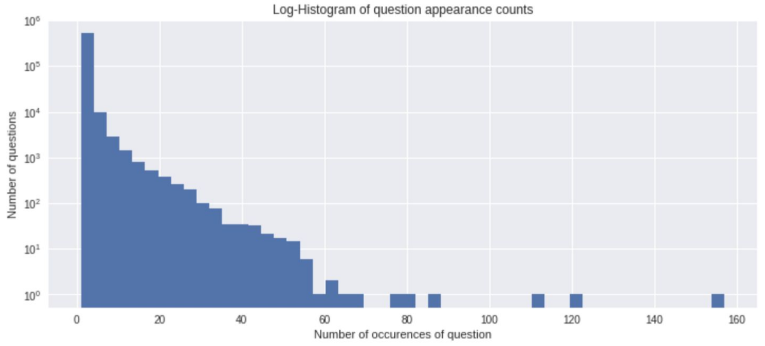
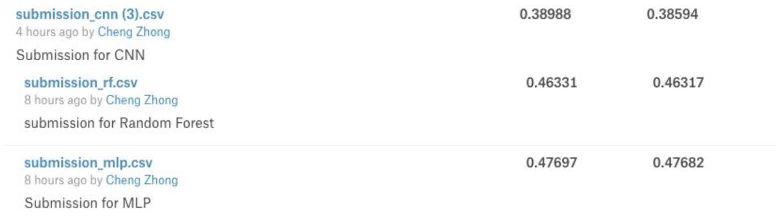


Figure 3. EDA on training data

```
Epoch 1/5
404290/404290 [==============================] - 256s 633us/step - loss: 0.4814 - acc: 0.7645
Epoch 2/5
404290/404290 [==============================] - 252s 624us/step - loss: 0.3631 - acc: 0.8335
Epoch 3/5
404290/404290 [==============================] - 252s 623us/step - loss: 0.2869 - acc: 0.8738
Epoch 4/5
404290/404290 [==============================] - 252s 623us/step - loss: 0.2301 - acc: 0.9017
Epoch 5/5
404290/404290 [==============================] - 252s 622us/step - loss: 0.1886 - acc: 0.9212
```

Figure 4. CNN result



submission_cnn (3).csv 4 hours ago by Cheng Zhong Submission for CNN	0.38988	0.38594
submission_rf.csv 8 hours ago by Cheng Zhong submission for Random Forest	0.46331	0.46317
submission_mlp.csv 8 hours ago by Cheng Zhong Submission for MLP	0.47697	0.47682

Figure 5. Kaggle Score Board Result

Discussion

As from the result, the CNN model is definitely outperforms the random forest model Quora is currently using. There are several reasons to choose the CNN model rather than LSTM:

1. The CNN model will not overfit as much as LSTM
2. The running time for CNN model is shorter than LSTM, for TESLA GPU on google colab, the running time is around 220s/epoch, as opposed to 400sec/epoch for a 250 unit LSTM

Further concerns on the CNN model:

1. The CNN model has no concept of word importance (such as TFIDF)
    a. Can be solved by add with a dense layer in our model
2. Will not distinguish sentence have same words but different word order

About the CNN Layer:

1. Choose the GlobalAveragePooling layer instead of max pooling is because it is faster and performs better