Project Title: Intent Based Duplicate Question Removal

Team Number: 14

Team Members:

Nikitha Rao (01FB15ECS364)

Rahul Ragesh (01FB15EEC303)

Abstract:

Quora, a well known social media platform, allows people from all around the world to ask questions and enables them to connect with various domain experts who provide quality answers and several new insights. With over 100 million people visiting Quora every month, the user base is humungous. It is not difficult to inference that several people may end us asking similarly worded questions. When there are multiple questions being asked with the same intent, a reader would have to spend more time in order to find the most suitable answer to their question. At the same time a writer may feel that they are answering multiple versions of the same question which may seem like a waste of time. This in turn leads us to our problem. Given any two questions, identify whether they have the same intent i.e. is one question a duplicate of the other or not. We will be using the dataset provided by Quora for the same. However, we aim to generalise our approach to be able to tackle the problem for any Question - Answer Forum, such as Quora itself or Stack Overflow.

Libraries Used:

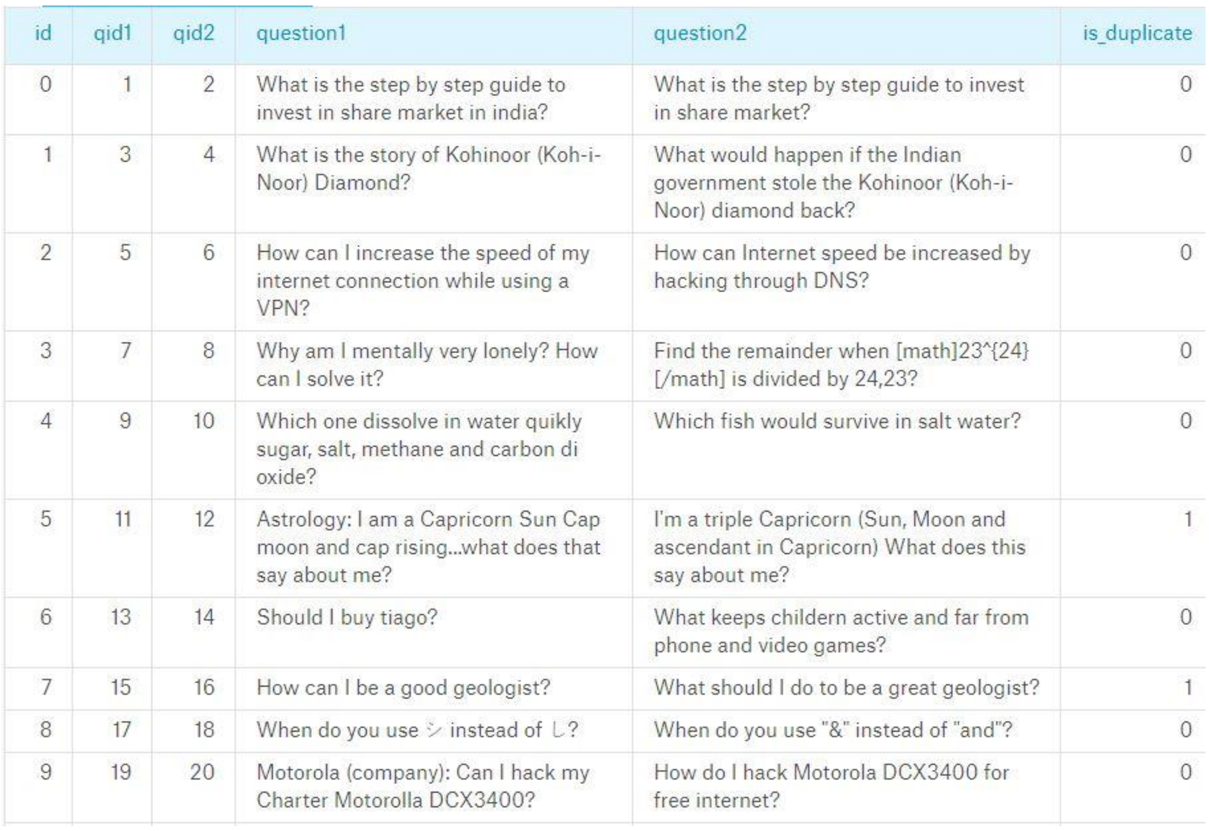
* numpy
* Tensorflow
* Keras
* scikit-learn
* h5py
* hdf5

Dataset Used:

Quora Question Pairs:

Link - https://data.quora.com/First-Quora-Dataset-Release-Question-Pairs

There are over 400,000 lines of potential question duplicate pairs. Each line contains IDs for each question in the pair, the full text for each question, and a binary value that indicates whether the line truly contains a duplicate pair.



id	qid1	qid2	question1	question2	is_duplicate
0	1	2	What is the step by step guide to invest in share market in india?	What is the step by step guide to invest in share market?	0
1	3	4	What is the story of Kohinoor (Koh-i-Noor) Diamond?	What would happen if the Indian government stole the Kohinoor (Koh-i-Noor) diamond back?	0
2	5	6	How can I increase the speed of my internet connection while using a VPN?	How can Internet speed be increased by hacking through DNS?	0
3	7	8	Why am I mentally very lonely? How can I solve it?	Find the remainder when [math]23^{24}[/math] is divided by 24,23?	0
4	9	10	Which one dissolve in water quikly sugar, salt, methane and carbon di oxide?	Which fish would survive in salt water?	0
5	11	12	Astrology: I am a Capricorn Sun Cap moon and cap rising...what does that say about me?	I'm a triple Capricorn (Sun, Moon and ascendant in Capricorn) What does this say about me?	1
6	13	14	Should I buy tiago?	What keeps childern active and far from phone and video games?	0
7	15	16	How can I be a good geologist?	What should I do to be a great geologist?	1
8	17	18	When do you use シ instead of し?	When do you use "&" instead of "and"?	0
9	19	20	Motorola (company): Can I hack my Charter Motorolla DCX3400?	How do I hack Motorola DCX3400 for free internet?	0

GloVe: Global Vectors for Word Representation

GloVe is an unsupervised learning algorithm for obtaining vector representations for words. Training is performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space.

Project Files and their Descriptions:

main.py – The main function. Calls the rest of the functions in the order of the files mentioned below.

get_dataset.py – The preprocessing function. It checks if the dataset(Quora Questions) is present, if not it downloads the necessary data from the links specified. The downloaded data is extracted, tokenized and encodes it. We then check if the GloVe data is present and download it if not. This has information about the word embeddings. Using the GloVe embeddings we create a new embedding matrix to denote the word similarity for all the words present in the Quora dataset. This is followed by saving the preprocessed data in the form of files – q1_train.npy, q2_train.npy, label_train.npy, word_embedding_matrix.npy.

model_def.py - The preprocessed data is split in the form of test data and training data (1:9) ratio. We then define the model architecture for the feature extraction and classification problem. Hyper parameters are tuned and the various layers are defined in the neural network. The output is in the form of 0 or 1 representing true or false depending on whether the two questions are convey the same intent or not. The loss function used is binary cross-entropy.

training.py – Trains the model based on the training data for 50 epochs. After each epoch, there is an early stopping criteria through cross validation is used. The weights for the epoch having the best validation accuracy is saved in a file.

testing.py – The model is tested by using the test data.

Instructions to run:

```
python3 main.py
```

Control Flow of the Program:

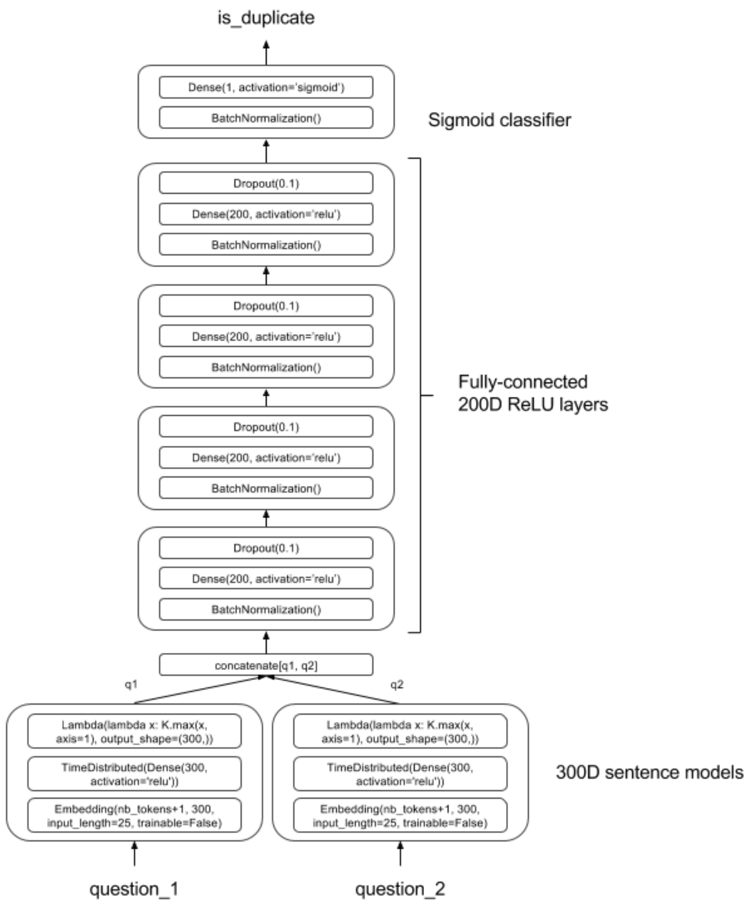
1. Pre-process the Dataset
2. Extract relevant information from the dataset.
3. Tokenize all the questions.
4. Encode text to sequence.
5. Generate word embedding matrix using Stanford GLoVe model.
6. Split the dataset into training, test and validation set.
7. Build the network model as per the given architecture.
8. Adam optimizer with Binary Cross Entropy Loss function was employed.
9. Save the configuration files.
10. Train the model.
11. Model evaluation is done to generate cross validation accuracy.

Training Time:

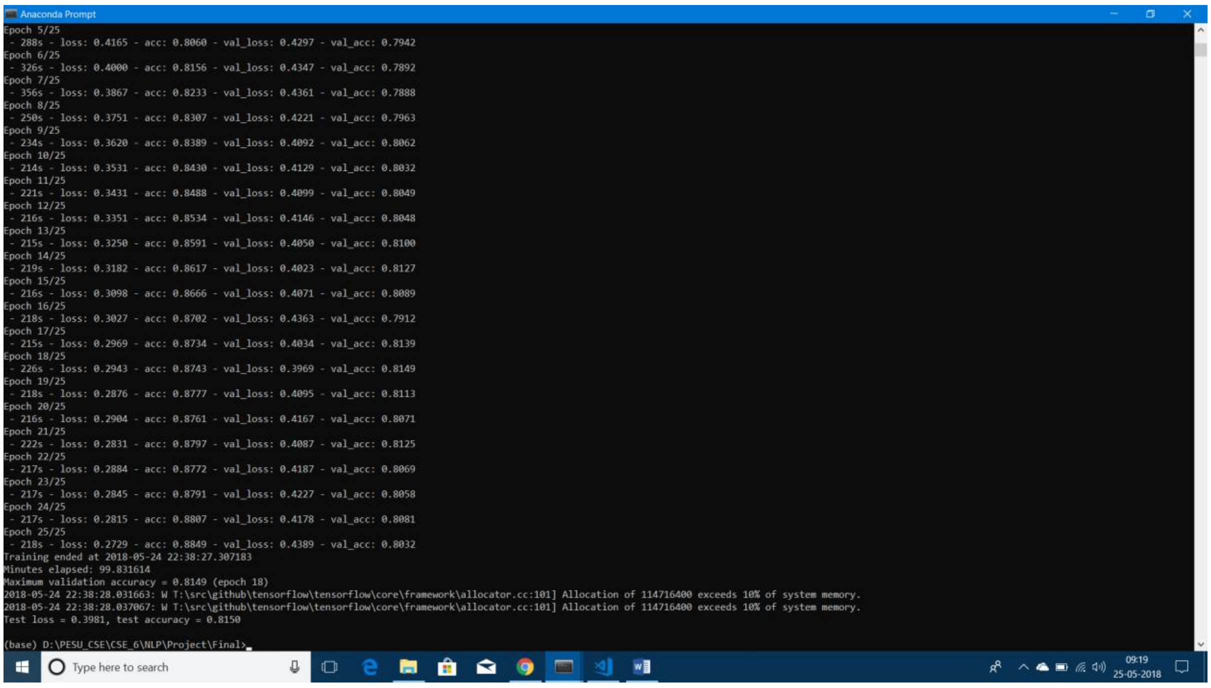
Training takes approximately 300 secs/epoch on an average, using Tensorflow as a backend for Keras on an intel core i7 8th gen, 8GB RAM. No GPU used.

For 50 epocs, 250 mins ~ 4 to 5 hours.

Network Architecture:


is_duplicate
Dense(1, activation='sigmoid')
BatchNormalization()
Sigmoid classifier
Dropout(0.1)
Dense(200, activation='relu')
BatchNormalization()
Fully-connected 200D ReLU layers
concatenate[q1, q2]
q1
q2
Lambda(lambda x: K.max(x, axis=1), output_shape=(300,))
TimeDistributed(Dense(300, activation='relu'))
Embedding(nb_tokens+1, 300, input_length=25, trainable=False)
300D sentence models
question_1
question_2

<u>Output:</u>



```
Epoch 5/25
 - 288s - loss: 0.4165 - acc: 0.8060 - val_loss: 0.4297 - val_acc: 0.7942
Epoch 6/25
 - 326s - loss: 0.4000 - acc: 0.8156 - val_loss: 0.4347 - val_acc: 0.7892
Epoch 7/25
 - 356s - loss: 0.3867 - acc: 0.8233 - val_loss: 0.4361 - val_acc: 0.7888
Epoch 8/25
 - 250s - loss: 0.3751 - acc: 0.8307 - val_loss: 0.4221 - val_acc: 0.7963
Epoch 9/25
 - 234s - loss: 0.3620 - acc: 0.8389 - val_loss: 0.4092 - val_acc: 0.8062
Epoch 10/25
 - 214s - loss: 0.3531 - acc: 0.8430 - val_loss: 0.4129 - val_acc: 0.8032
Epoch 11/25
 - 221s - loss: 0.3431 - acc: 0.8488 - val_loss: 0.4099 - val_acc: 0.8049
Epoch 12/25
 - 216s - loss: 0.3351 - acc: 0.8534 - val_loss: 0.4146 - val_acc: 0.8048
Epoch 13/25
 - 215s - loss: 0.3250 - acc: 0.8591 - val_loss: 0.4050 - val_acc: 0.8100
Epoch 14/25
 - 219s - loss: 0.3182 - acc: 0.8617 - val_loss: 0.4023 - val_acc: 0.8127
Epoch 15/25
 - 216s - loss: 0.3098 - acc: 0.8666 - val_loss: 0.4071 - val_acc: 0.8089
Epoch 16/25
 - 218s - loss: 0.3027 - acc: 0.8702 - val_loss: 0.4363 - val_acc: 0.7912
Epoch 17/25
 - 215s - loss: 0.2969 - acc: 0.8734 - val_loss: 0.4034 - val_acc: 0.8139
Epoch 18/25
 - 226s - loss: 0.2943 - acc: 0.8743 - val_loss: 0.3969 - val_acc: 0.8149
Epoch 19/25
 - 218s - loss: 0.2876 - acc: 0.8777 - val_loss: 0.4095 - val_acc: 0.8113
Epoch 20/25
 - 216s - loss: 0.2904 - acc: 0.8761 - val_loss: 0.4167 - val_acc: 0.8071
Epoch 21/25
 - 222s - loss: 0.2831 - acc: 0.8797 - val_loss: 0.4087 - val_acc: 0.8125
Epoch 22/25
 - 217s - loss: 0.2884 - acc: 0.8772 - val_loss: 0.4187 - val_acc: 0.8069
Epoch 23/25
 - 217s - loss: 0.2845 - acc: 0.8791 - val_loss: 0.4227 - val_acc: 0.8058
Epoch 24/25
 - 217s - loss: 0.2815 - acc: 0.8807 - val_loss: 0.4178 - val_acc: 0.8081
Epoch 25/25
 - 218s - loss: 0.2729 - acc: 0.8849 - val_loss: 0.4389 - val_acc: 0.8032
Training ended at 2018-05-24 22:38:27.307183
Minutes elapsed: 99.831614
Maximum validation accuracy = 0.8149 (epoch 18)
2018-05-24 22:38:28.031663: W T:\src\github\tensorflow\tensorflow\core\framework\allocator.cc:101] Allocation of 114716400 exceeds 10% of system memory.
2018-05-24 22:38:28.037067: W T:\src\github\tensorflow\tensorflow\core\framework\allocator.cc:101] Allocation of 114716400 exceeds 10% of system memory.
Test loss = 0.3981, test accuracy = 0.8150

(base) D:\PESU_CSE\CSE_6\NLP\Project\Final>
```