Learning of Semantically Equivalent Questions

Harini Shreedhar
has099@g.harvard.edu

Siva Rama Krishna Kottapalli
sik546@g.harvard.edu

Section 1: Abstract

Detecting duplicate questions or semantic equivalence between a pair of sentences is an active research area in NLP. New neural network architectures are being tested on this age-old problem to effectively identify the semantic equivalence between pair of sentences/questions.

Much of the motivation for choosing this topic is coming from the usefulness of resorting to Duplicate Question Detection (DQD) to support online question answering community forums, and also conversational interfaces, in general.

DQD falls under the broader task of semantic text similarity (STS). These challenges address a variety of STS subtasks, such as plagiarism detection, comparing machine translation output with a post-edited version, paraphrase detection, among several others. What these STS subtasks have in common is that, when given two input segments, the systems must rate their semantic similarity in some scale, ranging from total semantic equivalence to complete semantic dissimilarity.

For this project, the dataset was provided by Quora. We have implemented 2 models and resorted to a Siamese network in both these models. The first model is based on a Deep Convolution Neural Network and the second is based on a Bidirectional LSTM with Bahdanau Attention. In these two neural networks, various similarity measures such as Manhattan distance, Euclidean distance, Weighted Euclidean distance, Cosine Similarity etc. were used to determine the semantic equivalence between pairs of questions. We have also explored data augmentation technique to generate more positive samples for our project.

Dataset Link: Quora, Kaggle

YouTube Link: here

Section 2: Software Requirements (Windows Only)

A. Installing Python on Windows:

1. Install Python 3.5.X or higher version or Anaconda 2019.07
2. If the path is not already added to Environment variables you need to add following paths to windows path.
3. Go to My Computer -> Properties -> Advanced System Settings -> Environment Variables -> Environment Variables -> System Variables.
4. Append C:\Python3X to the PATH variable in System Variables.
5.Open command prompt and run python to see if windows can detect python

B. Installing Libraries on Windows:

Run following commands in the command prompt:

```
pip install os
pip install re
pip install json
pip install keras
pip install numpy
pip install pandas
pip install nltk
pip install scikit-learn
pip install 'tensorflow-gpu>=1.12,<1.13'/ tensorflow>=1.12,<1.13'
pip install time
pip install seaborn
pip install matplotlib
pip install jupyter
```

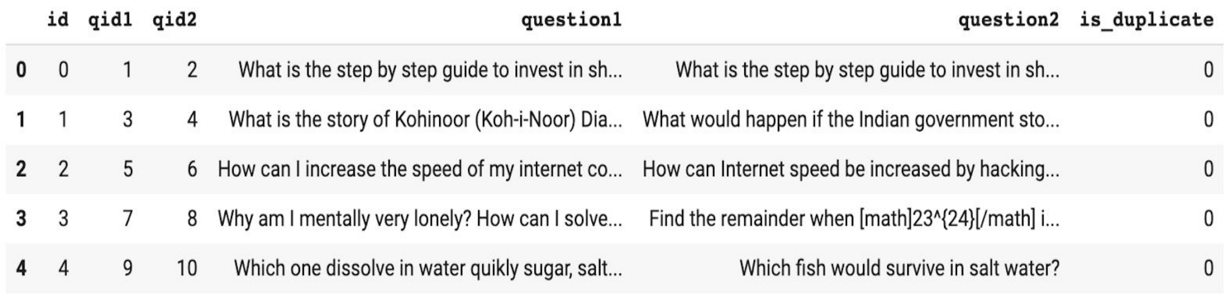
C. Installing CUDA Libraries for TensorFlow GPU version:

This is a very time-consuming process and hard to write down all the installation steps as these steps are very dependent on TensorFlow GPU version and operating system. We have tested the code using CUDA 9.0 and TensorFlow 1.12 version. Please use details in the link here to install CUDA.

Section 3: Datasets

A. Dataset extraction

Dataset is provided by Quora -- is an online platform where individuals or community of users who wish to seek an answer, covering a huge range of topics from politics to food to hobbies, computing etc, post a question. The introduced question is answered by volunteer experts. It contains a total number of 404,290 valid question pairs. The dataset is structured as following column labels: “id”, “qid1”, “qid2”, “question1”, “question2” and “is_duplicate”. Data points where is_duplicate value is 1, is considered to contain semantically equivalent question. Similarly, the test dataset contains totally 2,345,796 question pairs but without any “is_duplicate” label [1]. Figure 1 depicts a sample set of data points from the dataset.



	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: sample set of data points from the Quora dataset

B. Dataset description

The sampling method that was used to collect this dataset is reported as having returned an initially imbalanced dataset, with many more duplicates than non-duplicate pairs. Non-duplicate examples were subsequently added, including pairs of “related questions”. The dataset eventually released has 149,263 (37%) duplicate and 255,027 (63%) non-duplicate pairs [2] (see Figure 2). 13,698 questions appeared multiple times. It has also been mentioned that the dataset does contain some amount of noise and certain questions are synthetic. The possible estimation of noise is not provided. In order to obtain a neutral training, we used a balanced subset with equal number of duplicate and non-duplicate question pair in our training dataset.

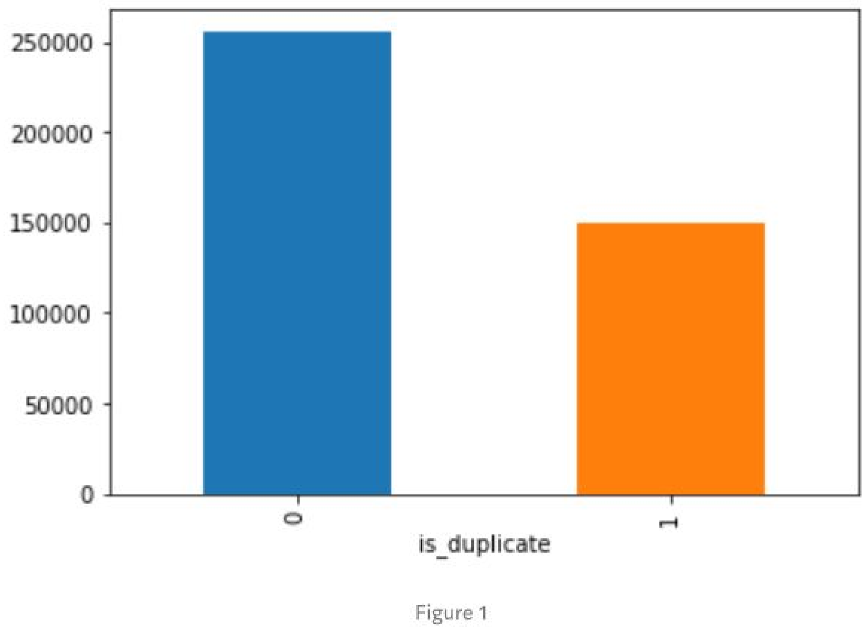


Figure 2: the number of duplicate and non-duplicate questions in the dataset

C. Dataset preparation

The Quora dataset provides a labeled dataset of pairs of questions. The question ID columns *qid1* and *qid2* were dropped as it is not useful for prediction. Next, we tokenized the entire dataset using the *nltk [3]* python package. For each question, all characters were converted to lowercase, numbers were removed, and special characters such as question marks and punctuation were removed.

For the LSTM model, certain other data preparations were added where words such as “I’m”, “we’ve” were changed to “I am”, “we have” respectively. This process is called as contraction process. Also, the stop words were removed from the text.

D. Word Embedding with GloVe - DCNN

The pre-trained *GloVe* model [4] was used to transform words to vectors. The vectorial representation of a word act as its projection into a continuous vector space. The position of a word within the vector space is learned from the text and is based on the words that surround the word when it is used.

Currently, the pre-trained Glove model is available with 50, 100 and 300 dimensions. Experimenting with different vector dimensions, we observed that the 300-dimensional representation had the best performance. Experiments detailed in this report use the 300-dimensional representation.

After tokenizing a question, each token was then replaced with its corresponding ID in the *GloVe* vocabulary using *one-hot encoding*.

The input questions vary significantly in length, from empty (0-length) up to 237 words. The average length of the sentence was found to be 59. In order to batch our computations during training and evaluation using matrix operations, we needed the input questions to all have a fixed length. To do so, we padded the shorter sentences at the end with a designated zero-padding. In our experiments we padded each question upto 300 in order to accomodate for future unseen question lengths.

Lastly, we split the dataset into training and validation. To have an unbiased training, the model was fed with equal number of duplicate and non-duplicate question pairs. For evaluating our models and tuning their parameters, the non-test dataset was randomly split into a training set and a validation set, containing 80% and 20% of the question pairs, respectively.

For the LSTM Model, we first split the data into train validation and test sets in 80:20 ratio. We again split train validation in the ratio of 80:20.

Section 4: Model Approaches

A. DCNN- Based Model

The pre-trained GloVe embeddings are sent through a fully connected, deep convoluted neural network (DCNN)[5][6].

The dataset comprises of pairs of questions and the objective of the model is to predict if they are semantically equivalent questions. Through the training process the model learns the probability distribution in all its classes. For this purpose, a Siamese architecture was opted. In the past, researchers have shown that Siamese network in detecting duplicate questions performs better than traditional SVM- based and Jaccard index[7]. Furthermore, the winner of the Kaggle competition on this task used a version of the Siamese network .

In the Siamese network, two similar sub-networks were architectured and the outputs from these two sub-networks were concatenated and forwarded to a dense layer. The final dense layer predicted the class of the input question pair.

Keras [8] and *Tensorflow [9]* was resorted for the implementation of the DCNN architecture. Figure 3 illustrates the outline of the network

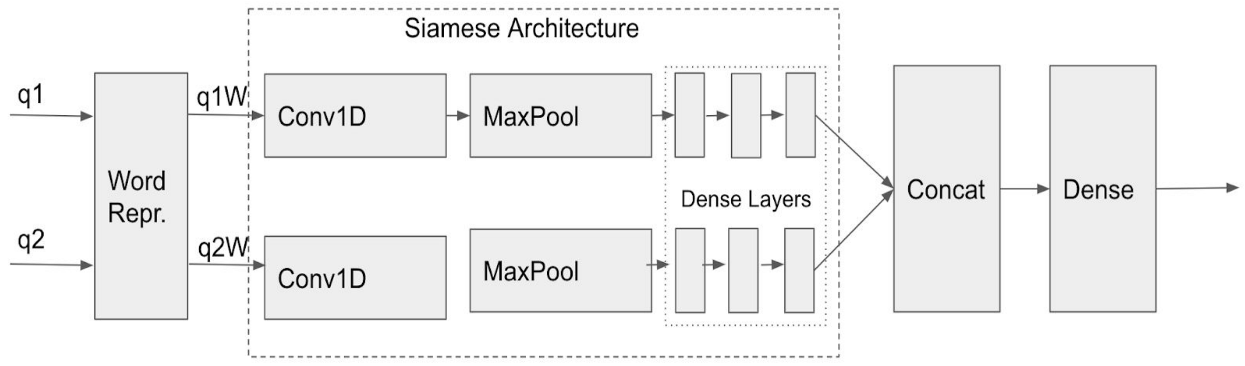


Figure 3**:** DCNN. WR- Word Representation; Conv1D - 1 Dimension Convolution Network; MaxPool- Max Pooling; Dense- Fully connected Dense layer; Concat - Concat Layer; Dense- Dense layer

After preparing the dataset, the input pairs were fed to the DCNN's sub layers. The first layer in the model represents the words in the vocabulary as a dense vectors using GloVe's *word2vec* representation.

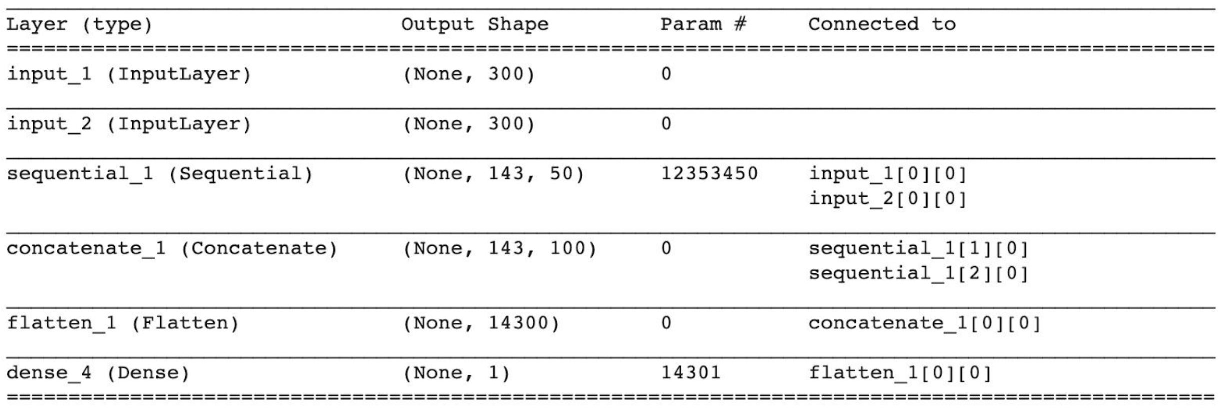
We instantiate the word representation layer as an embedding layer. The word embeddings are initialized using 300-dimensional GloVe vectors. The final word embeddings are learned through the training process, where during the back-propagation, the vector for each word gets adjusted and eventually determined given the neighboring words and the task at hand with which the cost function is computed.

During training, the DCNN model updates the word embeddings as well as the weights. After obtaining the vectorial representation of a question, it is 1st fed to the 1D convolution layer with 300 neurons and convolution window with kernel size of 15, followed by a max-pooling layer with size 2. The output of the max-pool layer is fed to 3, fully connected Dense layer with 50 neurons each. Finally, the output representations obtained from the 2 sublayers of the Siamese network is concatenated and fed to a Dense layer.

Inspired by the previous works by Chakaveh Saedi et al. [10][11], we initially tried cosine distance to predict if a pair was duplicate or not. Where, if the cosine is above an empirically determined threshold, the questions were said to be duplicates. However, we observed that the resulting model performed poorly in contrast to our concat model.

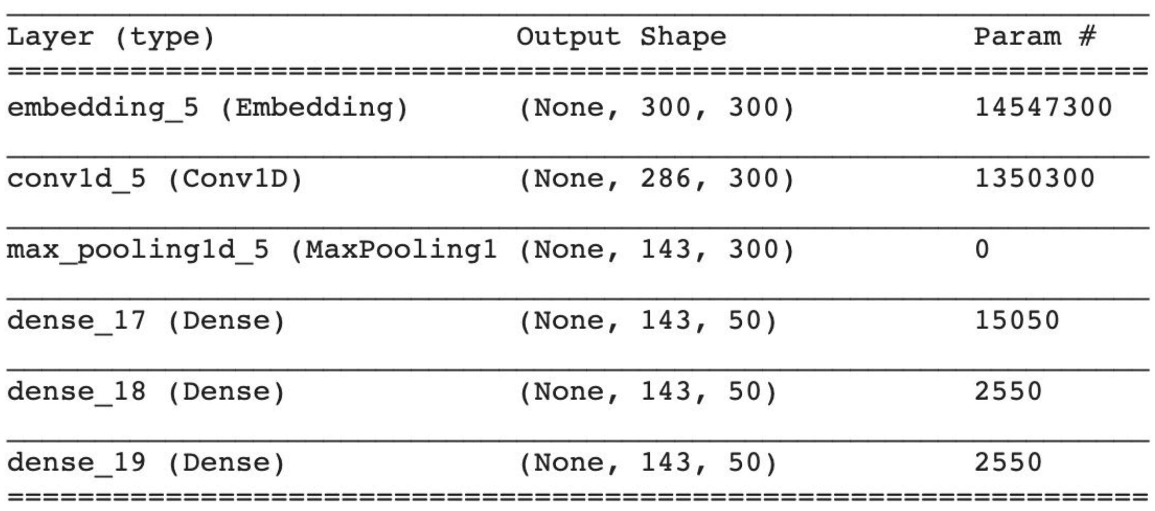
Through trial and error, we replaced the distance function with a neural network outputting a hyperbolic tangent over the two possible classes, leaving it up to this neural network to learn the correct distance function.

For every one of the experiments undertaken, with training datasets of different sizes, the learning was performed with a 0.01 learning rate, hyperbolic tangent as activation function and a stochastic gradient to compute the cost function with a mean squared error loss. In each experiment, the best score was picked from a twenty epoch training.



```
____________________________________________________________________________________________________
Layer (type)                     Output Shape          Param #     Connected to
====================================================================================================
input_1 (InputLayer)             (None, 300)           0
____________________________________________________________________________________________________
input_2 (InputLayer)             (None, 300)           0
____________________________________________________________________________________________________
sequential_1 (Sequential)        (None, 143, 50)       12353450    input_1[0][0]
                                                                   input_2[0][0]
____________________________________________________________________________________________________
concatenate_1 (Concatenate)      (None, 143, 100)      0           sequential_1[1][0]
                                                                   sequential_1[2][0]
____________________________________________________________________________________________________
flatten_1 (Flatten)              (None, 14300)         0           concatenate_1[0][0]
____________________________________________________________________________________________________
dense_4 (Dense)                  (None, 1)             14301       flatten_1[0][0]
====================================================================================================
```

Figure 4: DCNN model’s layer details



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_5 (Embedding)      (None, 300, 300)          14547300
_________________________________________________________________
conv1d_5 (Conv1D)            (None, 286, 300)          1350300
_________________________________________________________________
max_pooling1d_5 (MaxPooling1 (None, 143, 300)          0
_________________________________________________________________
dense_17 (Dense)             (None, 143, 50)           15050
_________________________________________________________________
dense_18 (Dense)             (None, 143, 50)           2550
_________________________________________________________________
dense_19 (Dense)             (None, 143, 50)           2550
=================================================================
```

Figure 5: DCNN model’s sub network’s layer details

B. LSTM- Based Model

We implemented LSTM models on Quora question pair dataset and this dataset can be downloaded from Quora website or Kaggle website. We have provided the links below. We have achieved an accuracy of 89% with GRU model with Bahdanau Attention when our models used Stanford GLOVE models which were built on 840B tokens, 2.2M vocab, 300d vectors. We did a little cleaning in the text as we are more interested in learning implementing the neural network. Main challenges we encountered while building this network are keeping up with network dimensions in different layers of the network. Building custom losses or implementing custom layers is difficult when you don’t keep track of the dimensions in different layers of the network. We were able to implement 4 different loses but the main loss which was implemented in the paper couldn’t be implemented as we keep on receiving multiple errors during training. We have included this loss, but it is commented out.

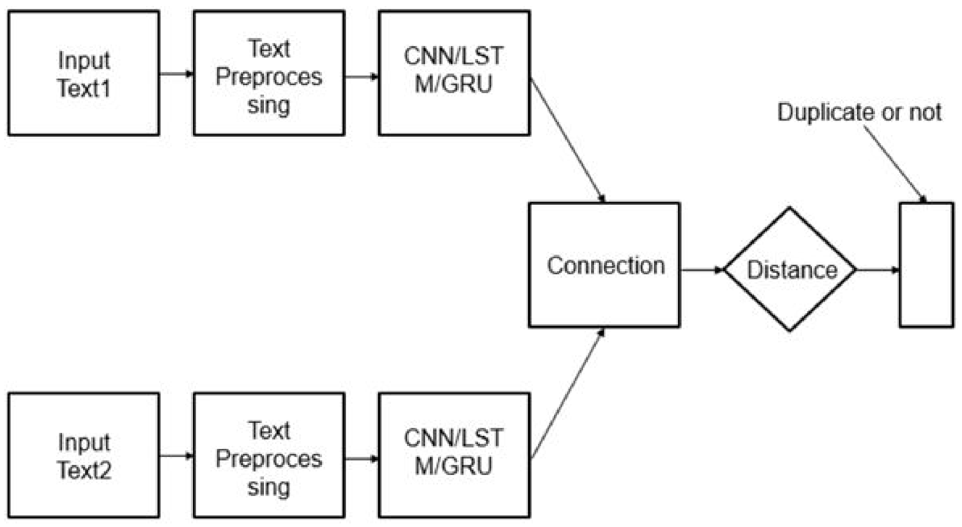


Figure 4: Deep Recurrent Neural Network with LSTM and GRU implementation

LSTM siamese network takes in two inputs that share same weights to compute the similarity coefficient between the two input vectors.

We created network with two types of main RNN cells. One is Gated Recurrent Units (GRU) and Long Short-Term Memory (LSTM) cells. We tested both unidirectional and bidirectional cells to see the performance of the model. In each recurrent unit we have added regularization to reduce overfitting of the model. Base Siamese network is created with single layer RNN units and two layers RNN units. To reduce overfitting of the model, we added dropout layer. We added Bahdanau attention to further improve performance of the model.

Time distributed layer is added to model to unroll the output of the RNN units for further processing before applying attention. Then flatten layer is added to reshape the vector output. “Tanh” activation is applied in time distributed layer to start attention process.

1. LSTM models tested:

1. Single Layer GRU
2. Single Layer LSTM
3. Two Layer GRU
4. Two Layer LSTM
5. Single Layer Bidirectional LSTM
6. Two Layer Bidirectional LSTM

In Bidirectional LSTM, we need to concatenate the output from each direction of LSTM. After creating the base network, we pass input1 and input2 for question1 and question2 to the loss function. This creates the loss and then we concatenate all Siamese output1 and Siamese output2 and loss. Then we slowly tried to reduce the number of parameters in the by creating a network similar in shape of funnel which finally outputs duplicate or not.

2. Loss Functions:

As shown in the paper, we tried to create four different loss functions.

1. Euclidean Distance
2. Manhattan Distance
3. Weighted Manhattan Distance
4. Cosine Similarity
5. Custom SoftMax Loss Function*

* Tried creating Custom SoftMax Loss function which showed superior results when compared to other loss functions. But we couldn’t successfully write code for this loss function due to

input/output shape compatibility issues. We used **Weighted Manhattan Distance** in all our models

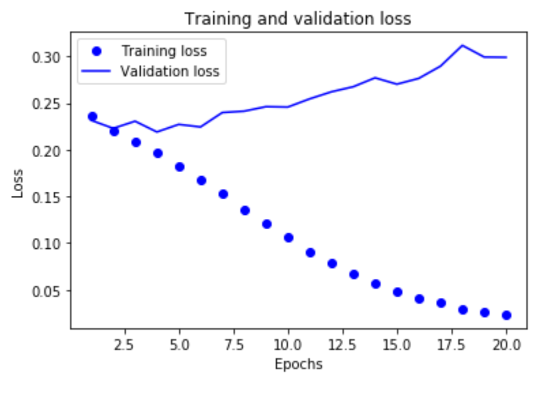
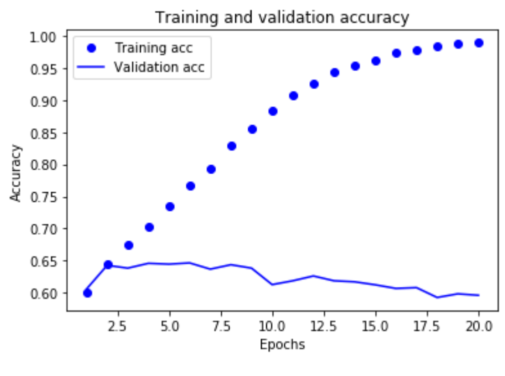
Section 5: Results

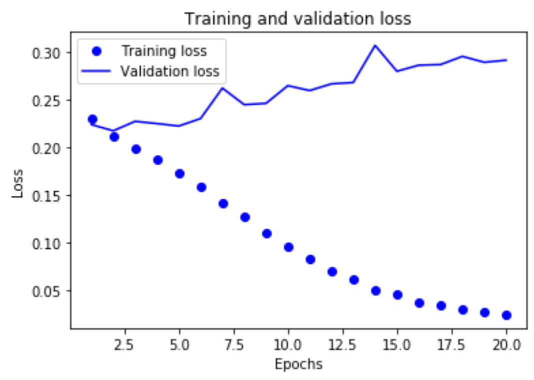
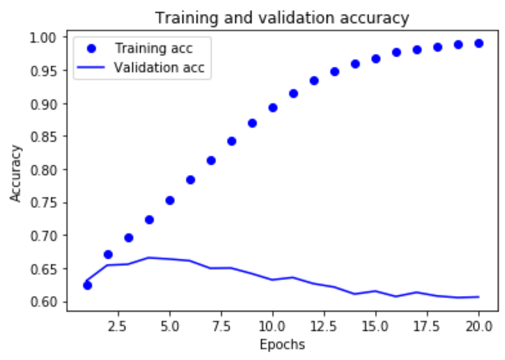
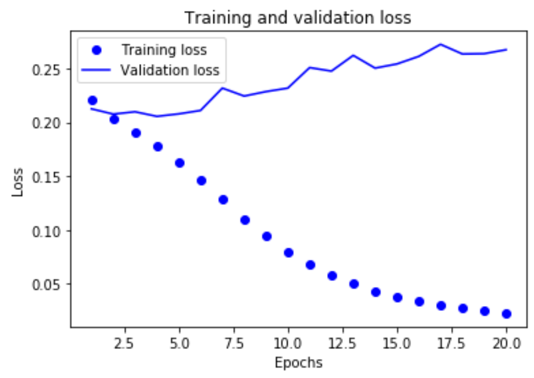
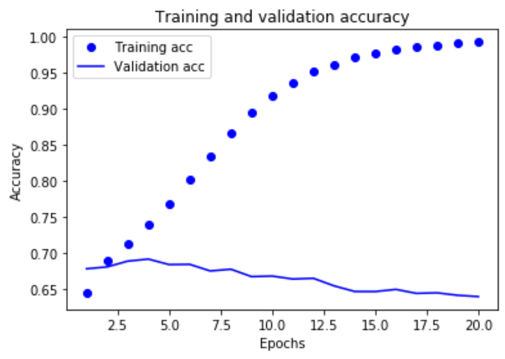
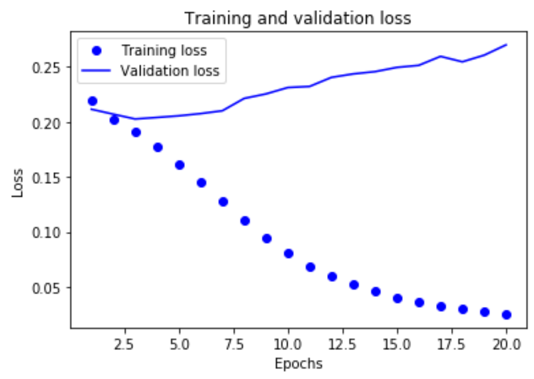
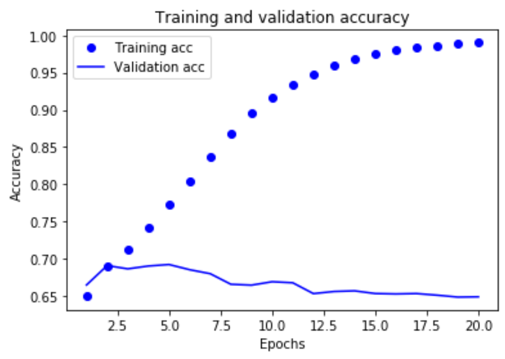
A. DCNN- Based Model

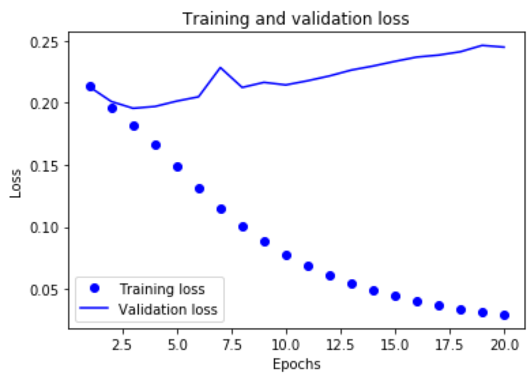
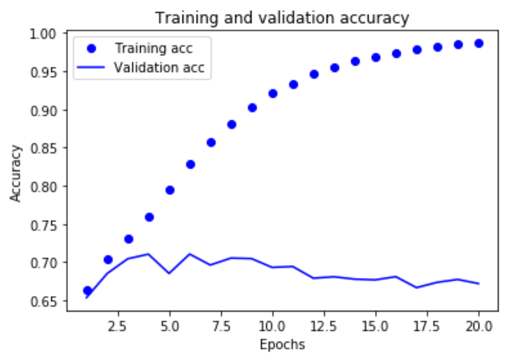
Table 1: Number of training parameters and accuracy of the DCNN model under different training set sizes. The model was trained using Colaboratory on a GPU. The time taken for training each of the subsets was 8mins, 25mins, 40 mins, 50 mins, 1.2hrs for 8k, 20k, 65k, 80k and 140k pairs respectively.

DCNN	**# 8k pairs**	**# 20k pairs**	**# 65k pairs**	**# 80k pairs**	**# 140k pairs**
# Training params	5,921,750	5,927,150	14,555,150	15,917,750	20,426,150
Accuracy	64.66%	66.55%	69.17%	69.23%	**71.04%**

Table 2: Plots of Loss and Accuracy for the runs in Table 2

# of pairs	**Loss**	**Accuracy**
8K		

20k	Training and validation loss Training loss Validation loss Loss Epochs	Training and validation accuracy Training acc Validation acc Accuracy Epochs
65K	Training and validation loss Training loss Validation loss Loss Epochs	Training and validation accuracy Training acc Validation acc Accuracy Epochs
80K	Training and validation loss Training loss Validation loss Loss Epochs	Training and validation accuracy Training acc Validation acc Accuracy Epochs

140k		

B. LSTM Based Model

Model	Accuracy	Precision	Recall	F1 Score
Single Layer GRU	88 %	0.86	0.96	**0.91**
Single Layer LSTM	88 %	0.92	0.89	0.90
Two Layer GRU	**89** %	0.88	0.96	**0.91**
Two Layer LSTM	88 %	0.88	0.94	**0.91**
Single Layer Bidirectional LSTM	85 %	0.82	**0.99**	0.89
Two Layer Bidirectional LSTM	88 %	**0.93**	0.88	0.90

Table 3::

Sections 6: Conclusion

Through experiments we observed that the validation accuracy increases with increase in training size. We observed a significant improvement upto 65k pairs (~4% increase) after which there was only a marginal improvement in accuracy. From Table 3 it can be observed that due to overfitting, the validation accuracy increases during the initial epocs and subsequently decreases. The cosine loss function and softmax did not provide good results on validation accuracy, as a result a concat layer was implemented for the DCNN.

A. What didn’t work:

We didn’t work much on cleaning the data. Performance would have been better if we had followed word splitting technique similar to BERT model. For the LSTM, custom SoftMax loss function didn’t work due to the shape of the input vectors. There is a shape mismatch between some of the layers in this loss function. Didn’t work on finding optimal hyper parameters. We just created models with minimum number of parameters so that the models can be run in sufficient time on lower powered hardware.

Section 7: References:

[1] Dhakal, Ashwin, Arpan Poudel, Sagar Pandey, Sagar Gaire, and Hari Prasad Baral. "Exploring Deep Learning in Semantic Question Matching." In 2018 IEEE 3rd International Conference on Computing, Communication and Security (ICCCS), pp. 86-91. IEEE, 2018.
[2] Iyer, Shankar, Nikhil Dandekar, and Kornél Csernai. "First quora dataset release: Question pairs." data. quora. com(2017).
[3] Bird, Steven, Ewan Klein, and Edward Loper. Natural language processing with Python: analyzing text with the natural language toolkit. " O'Reilly Media, Inc.", 2009.
[4] Pennington, Jeffrey, Richard Socher, and Christopher Manning. "Glove: Global vectors for word representation." In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), pp. 1532-1543. 2014.
[5] Bogdanova, Dasha, Cicero dos Santos, Luciano Barbosa, and Bianca Zadrozny. "Detecting semantically equivalent questions in online user forums." In Proceedings of the Nineteenth Conference on Computational Natural Language Learning, pp. 123-131. 2015.
[6] Afzal, Naveed, Yanshan Wang, and Hongfang Liu. "Mayonlp at semeval-2016 task 1: Semantic textual similarity based on lexical semantic net and deep learning semantic model." In Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016), pp. 674-679. 2016.
[7] Wu, Yan, Qi Zhang, and Xuanjing Huang. "Efficient near-duplicate detection for q&a forum." In Proceedings of the 5th International Joint Conference on Natural Language Processing, pp. 1001-1009. 2011.
[8] Gulli, Antonio, and Sujit Pal. Deep Learning with Keras. Packt Publishing Ltd, 2017.

[9] Abadi, Martín, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado et al. "Tensorflow: Large-scale machine learning on heterogeneous distributed systems." arXiv preprint arXiv:1603.04467 (2016).
[10] Saedi, Chakaveh, Joao Rodrigues, João Silva, and Vladislav Maraev. "Learning profiles in duplicate question detection." In 2017 IEEE International Conference on Information Reuse and Integration (IRI), pp. 544-550. IEEE, 2017.

[11] Homma, Yushi, Stuart Sy, and Christopher Yeh. "Detecting duplicate questions with deep learning." In Proceedings of the 30th Conference on Neural Information Processing Systems (NIPS 2016). IEEE, pp. 1-8. 2016
[12] https://www.quora.com/q/quoradata/First-Quora-Dataset-Release-Question-Pairs
[13] https://www.kaggle.com/c/quora-question-pairs
[14] https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1174/reports/2748045.pdf
[15] https://ieeexplore.ieee.org/document/8102981
[16] https://nlp.stanford.edu/projects/glove/
[17] https://stackoverflow.com/questions/43018030/replace-apostrophe-short-words-in-python
[18]https://towardsdatascience.com/how-to-implement-seq2seq-lstm-model-in-keras-shortcutnlp-6f355f3e5639
[19] https://github.com/keras-team/keras/blob/master/examples/mnist_siamese.py
[20]https://github.com/LuJunru/Sentences_Pair_Similarity_Calculation_Siamese_LSTM/blob/master/train.py
[21]https://datascience.stackexchange.com/questions/45165/how-to-get-accuracy-f1-precision-and-recall-for-a-keras-model
[22] https://github.com/keras-team/keras/issues/2115
[23]https://stackoverflow.com/questions/57190848/concatenate-multiple-lstm-outputs-of-shape-none-m-in-keras-custom-layer
[26]https://stackoverflow.com/questions/47877475/keras-tensorboard-plot-train-and-validation-scalars-in-a-same-figure?rq=1