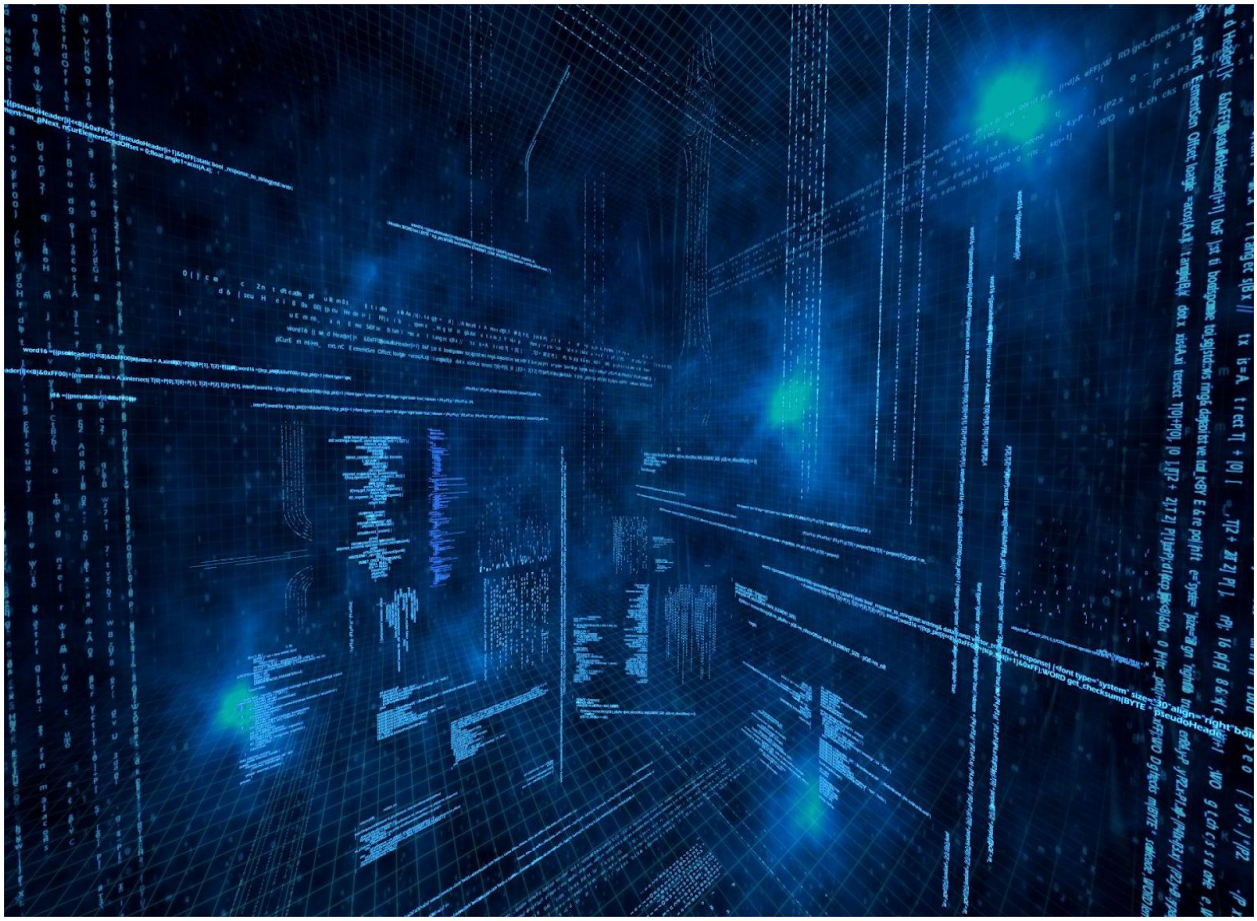
Project Report

Lab Exam 2

Detecting similarity in Quora questions



Sumeet Padavala	01FB15ECS315
Ankit Anand	01FB15ECS041
Abhijay Gupta	01FB15ECS005

INTRODUCTION

This assignment used a dataset from a well-known Kaggle competition created from data sampled from Quora.

The goal of the competition was to detect duplicate questions among randomly sampled pairs of questions.

In this document, we describe our solution to this challenge using a custom Keras model and pre-trained GloVe vector embeddings.

DATA PROCESSING

We collected data from the provided JNResearch API. We collected a total of 100,000 samples, though we only used about 15,000 samples due to compute resource constraints.

The data was provided as a list of dictionaries containing the text of both questions, instance id, ids for both questions, and the target column is_duplicate.

We first converted the data to JSON and then processed it through pandas to convert it to a DataFrame.

We then removed the unnecessary columns, leaving two columns for the question text, and one for the target column.

To preprocess the question text, we used NLTK to word-tokenize the sentence, padded the sequences to a constant length, and then mapped each sentence to an index using a precomputed mapping generated from the GloVe embeddings.

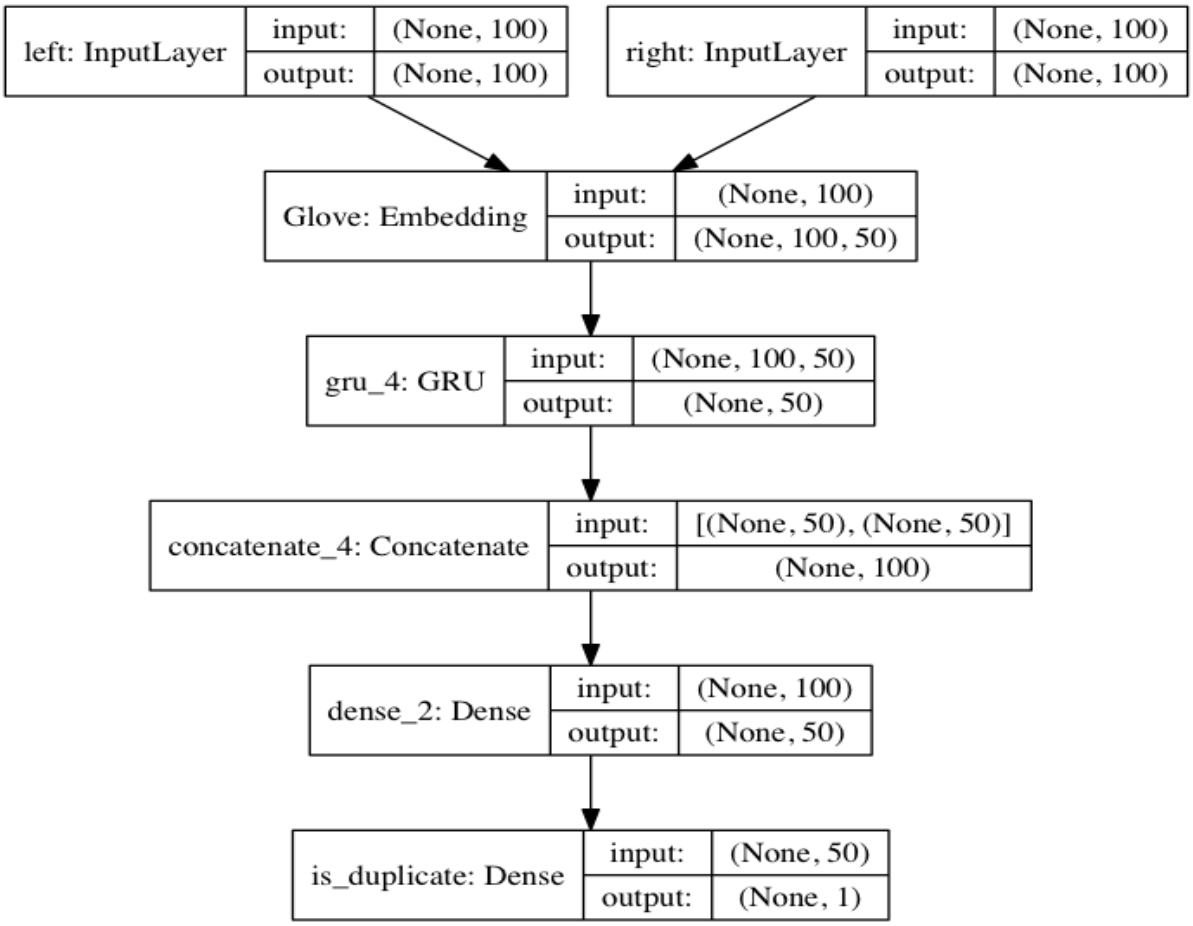
We then split the data into training and testing sets with an 85/15 ratio.

Finally, we extracted this data to numpy arrays, after careful manipulation to ensure the shape of the data correctly mapped to the model's input shape.

MODEL

Our model derives from the concept of Siamese networks, which have been successfully utilized for measuring the similarity between inputs for a variety of input types.

1. The model expects two sequences of inputs to be passed to each of the two Input layers.
2. The input sequences are processed by a shared embedding layer preloaded with the 50-dimensional GloVe embedding set.
3. This sequence of word embeddings is fed to a (shared) GRU layer for initial processing.
4. The final outputs from the GRU layer for both input sequences are concatenated into one 100-dimensional vector and fed to a final feedforward network. In our case, we have utilized two dense layers, with the final layer producing a single sigmoid-activated output.



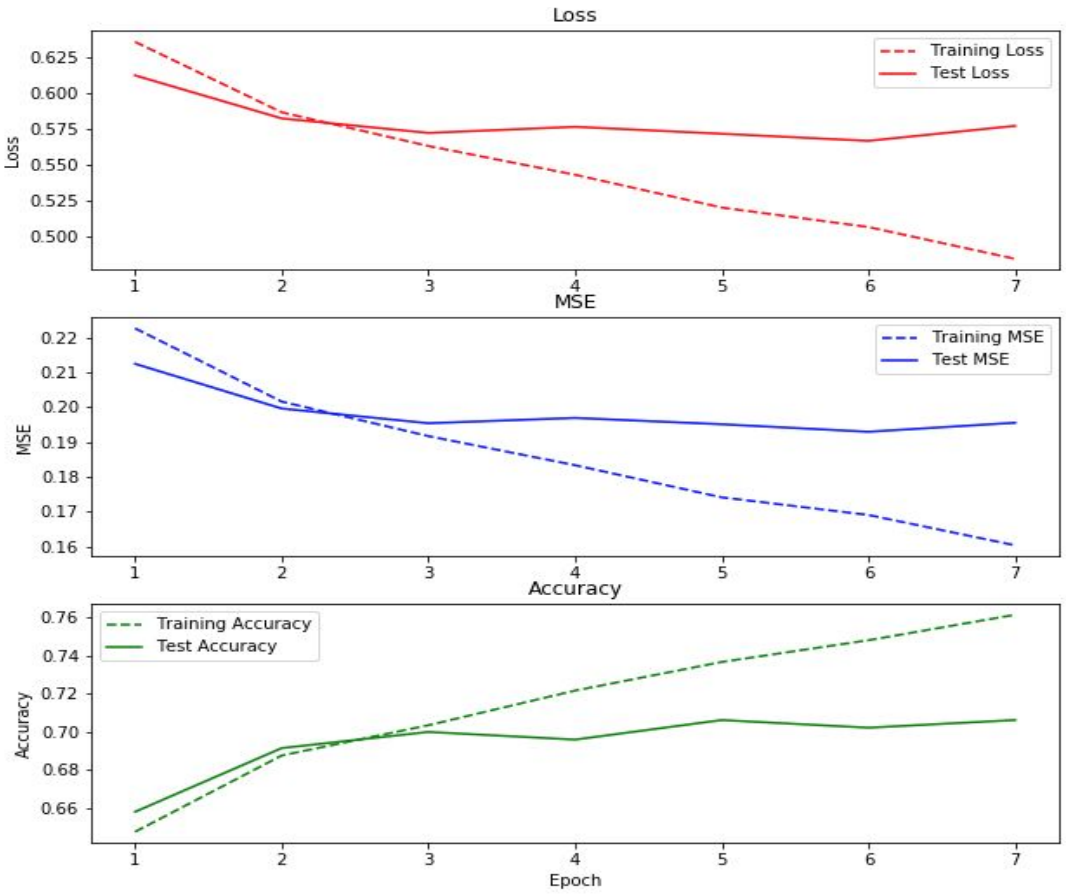
ANALYSIS

We trained the model for seven epochs on 12,750 input samples, with 2250 samples for validation data. The model utilized the standard Adam optimizer with default (Keras-provided) parameters. The model parameters are as follows:

1. Left Input and Right Input Layer:- 50-dimensional index sequence
2. Embedding Layer (Left and Right):- 50-dimensional GloVe vectors
3. GRU:- 50 Hidden Units
4. Dense layer:- 50 Neurons with Relu as an activation
5. Output layer:- 1 Neuron with Sigmoid activation

Our final validation accuracy fluctuates between 65% and 70% between runs.

Some statistics are represented in this chart:



As the charts show, our model begins to overfit quite quickly, usually after the third or fourth epoch. Both parameter tuning and larger datasets(50,000 samples) seem to have little effect, which leads us to believe that we have found the best local maxima for our model and approach.

We have hypothesized a more complex model that may produce better accuracy, however, we were unable to implement it within the assignment deadline.

CHALLENGES FACED

1. We originally had some trouble loading and processing the data from the given dictionary format to something that could be processed by our model. We eventually used the admittedly suboptimal approach of using JSON and DataFrames as an intermediate. This process is very CPU-heavy and will not scale to larger datasets and particularly online learning.
2. We had difficulty in effectively testing more complex models since larger models and datasets tended to crash our systems. As a result, we have used a very simple model (three trainable layers) and trained it with just 15,000 of our 100,000 collected samples.
3. We were unable to produce a model with a validation accuracy greater than 70%, and we believe that achieving a better score is impossible without a more complex model.

REFERENCES

1. Keras documentation (keras.io)
2. Tensorflow Keras documentation (tensorflow.org/api_docs/python/tf/keras)
3. Matplotlib reference (matplotlib.org/api/pyplot_api.html)