Semantic Text Similarity
- Detecting Duplicate Questions

Submitted To -
Dr. Manish Shrivastava

Submitted By -
Dhawnit Khurana (20162076)
Hemant Verma (20162078)
Kanishtha Surana (20162080)

Project Overview -
In this project we determine semantic equivalence between pairs of questions using Deep Learning (Siamese Bidirectional Lstm) on a dataset released by Quora.

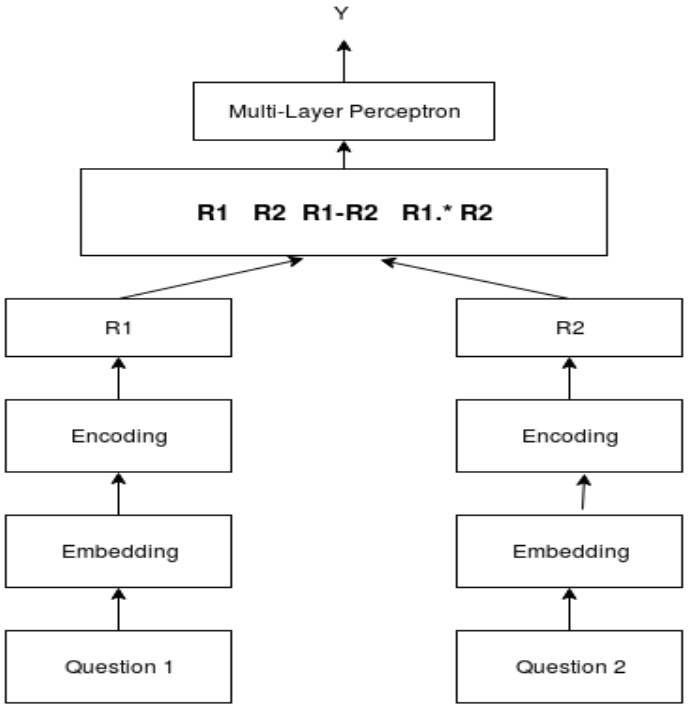
Need -
Quora provides users with a platform to ask questions that other users on the site may answer. However, many of the questions being asked at any given time have already been asked by other users, usually with different wording or phrasing. Ideally, these duplicate questions should be merged together into a single canonical question, as doing so would provide a number of benefits:

- It saves the question asker time if their question has already been answered previously on the site.
- Q&A knowledge bases have more value to users and researchers when there is a single canonical question and collections of answers, instead of having the knowledge fragmented and spread throughout the site.
- Having knowledge of alternative phrasings of the same question can improve search and discovery.

Approach used -

Deep Learning approach used is a Siamese Birectional LSTM network with a Multilayer Perceptron at the output layer.Siamese networks seem to perform well on similarity tasks and have been used for tasks like sentence semantic similarity, recognizing forged signatures and many more.

Architecture -



Project Modules -

- **Preprocessing -** We get the data as raw text. Then we take each question, remove all the stop words and the words not present in Word2Vec from it. And also applied some of the tokenization regex's to clean the data. And then we convert the questiont into lists of word indices. Then we fill the Embedding matrix based on our Vocabulary. We used Word2Vec Embeddings for the words. We also found the maximum length of the questions (maxSeqLength). And based on the maxSeqLength, padded the smaller length questions with 0.

- **Data Splitting -** There were 404,351 question pairs. Out of this we kept aside 40,000 question pairs as Validation set. And 20,000 question pairs as Test set.

- **Siamese LSTM Model with Multi-Layer Perceptron at the Output** - First we have a Input Layer. Input layer does nothing except defining the shape of the input data (maxSeqLength) to our model. In fact it creates a tensor that is used as input to the other layers. Next we have the Embedding Layer which takes input as maxSeqLength Vector, and using Embedding matrix produces a sentence embedding. Then we have the Bidirectional LSTM Layer which is shared between two siamese network components. We took 50 hidden units in the LSTM Layer. The output of this layer is 50 dimensional vector. Then we form a new vector out of this output of the form as shown above in the architecture. Reason for forming such a

vector instead of computing some distance (Cosine, Euclidean and Manhattan) over the outputs of the 2 LSTM components is we do not know what a “natural” distance measure is in the sentence vector space. To address this problem, we replaced the distance function with a neural network outputting a softmax over the two possible classes, leaving it up to this neural network to learn the correct distance function.

- **Model Evaluation** - Evaluated the model on 20,000 question pairs.

Results -
Validation set Accuracy - 82.59%
Test Set Accuracy - 82.09%

```
hemant@hemantpc:~/Dropbox/nlpProject$ python nlpProj.py
Using TensorFlow backend.
2017-11-21 01:33:39.216066: W tensorflow/core/platform/cpu_feature_guard.cc:45] The TensorFlow library wasn't compiled to use SSE
4.1 instructions, but these are available on your machine and could speed up CPU computations.
2017-11-21 01:33:39.216108: W tensorflow/core/platform/cpu_feature_guard.cc:45] The TensorFlow library wasn't compiled to use SSE
4.2 instructions, but these are available on your machine and could speed up CPU computations.
2017-11-21 01:33:39.216125: W tensorflow/core/platform/cpu_feature_guard.cc:45] The TensorFlow library wasn't compiled to use AVX
 instructions, but these are available on your machine and could speed up CPU computations.
nlpProj.py:206: UserWarning: Update your `Model` call to the Keras 2 API: `Model(inputs=[<tf.Tenso..., outputs=Tensor("de...)`
  model = Model(input=[left_input,right_input], output=preds)
Train on 344351 samples, validate on 40000 samples
Epoch 1/5
344351/344351 [==============================] - 11864s - loss: 0.1707 - acc: 0.7493 - val_loss: 0.1481 - val_acc: 0.7832
Epoch 2/5
344351/344351 [==============================] - 11867s - loss: 0.1423 - acc: 0.7942 - val_loss: 0.1391 - val_acc: 0.7982
Epoch 3/5
344351/344351 [==============================] - 11968s - loss: 0.1313 - acc: 0.8121 - val_loss: 0.1299 - val_acc: 0.8149
Epoch 4/5
344351/344351 [==============================] - 12504s - loss: 0.1244 - acc: 0.8239 - val_loss: 0.1259 - val_acc: 0.8206
Epoch 5/5
344351/344351 [==============================] - 12127s - loss: 0.1191 - acc: 0.8322 - val_loss: 0.1239 - val_acc: 0.8259
Training time finished.
5 epochs in 16:45:36.718199
acc: 82.09%
Saved model to disk
hemant@hemantpc:~/Dropbox/nlpProject$ ~
```