Quora questions similarity

Konstantin Lekomtsev
Data Science Career Track

Mentor: Nishant Gupta

Objective

The main objective of the project was to identify duplicate question pairs in Quora question pairs dataset from Kaggle [1].

“Quora is a place to gain and share knowledge—about anything. It’s a platform to ask questions and connect with people who contribute unique insights and quality answers. This empowers people to learn from each other and to better understand the world. Multiple questions with the same intent can cause seekers to spend more time finding the best answer to their question, and make writers feel they need to answer multiple versions of the same question. Quora values canonical questions because they provide a better experience to active seekers and writers, and offer more value to both of these groups in the long term.”[1]

Dataset

The dataset was downloaded from Kaggle. Key statistics:

- 404,278 question pairs,
- 37% of duplicate reviews.
- Maximum question length, question 1: 125
- Maximum question length, question 2: 237
- Median number of words per question: 10
- Vocabulary size: 232531 words

“The ground truth is the set of labels that have been supplied by human experts. The ground truth labels are inherently subjective, as the true meaning of sentences can never be known with certainty. Human labeling is also a 'noisy' process, and reasonable people will disagree. As a result, the ground truth labels on this dataset should be taken to be 'informed' but not 100% accurate, and may include incorrect labeling. We believe the labels, on the whole, to represent a reasonable consensus, but this may often not be true on a case by case basis for individual items in the dataset.”[2]

Preprocessing and feature engineering

As a first step Jaccard (on words and characters), cosine and Manhattan similarity metrics were added as features in the dataset. The Jaccard similarity index compares members for two sets to see which members are shared and which are distinct. Cosine similarity is a measure of similarity between two non-zero vectors of an inner product space that measures the cosine of the angle between them. Manhattan distance is the distance between two points measured along axes at right angles. In a plane with p1 at (x1, y1) and p2 at (x2, y2), it is |x1 - x2| + |y1 - y2|.

The distribution of similarity metrics in all question pairs is shown in Fig. 1. These distributions are very different, even though they represent the same corpus of text. For example, Jaccard similarity on characters shows that the question pairs are largely similar, which is an intuitive result since there is a fixed number of characters in the English alphabet. As for the words, the cosine similarity is almost uniform and Jaccard similarity is skewed towards lover similarity of questions in the dataset.

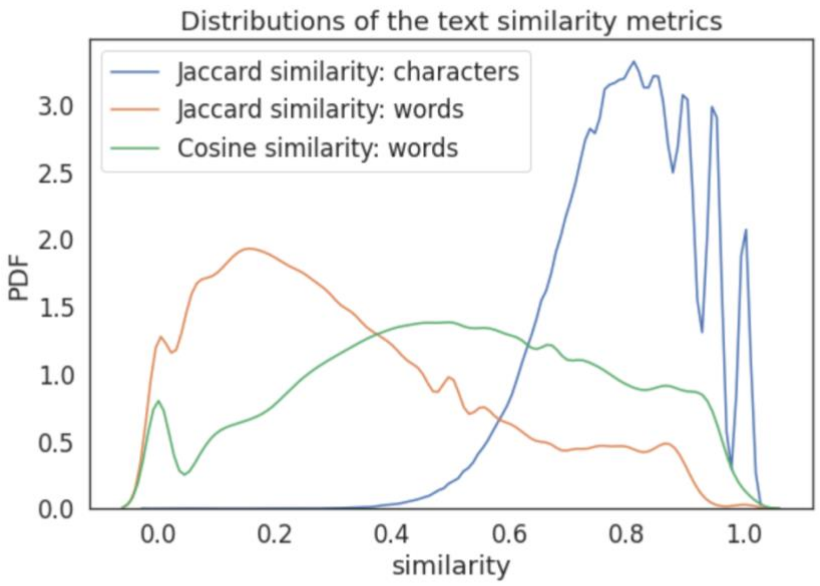

Fig. 1 The distribution of similarity metrics in all question pairs.

The following pre-processing workflow was applied:

- Tokenize dataset.
- Pad sequences. Limit the number of words considered per question to a certain number (20 in our case due to limited computational resources, free GPU on google colab was used).
- Generate an embedding matrix using pre-trained embeddings using GloVe with 50 dimensions per word.

Modeling

Two architectures were applied to analyze question similarity.

-> One-shot learning using a twin neural network with time distributed layers, and cosine and Jaccard similarity (time distributed network).

-> One-shot learning using a twin neural network with shared LSTM and Manhattan similarity (LSTM network).

For the time distributed network:

- Add embedding layers and then time distributed layers for each question.
- Use previously calculated similarity metrics as input features: cosine similarity on words, Jaccard similarity on characters and words.
- Add features equivalent to finding the most similar words in question pairs (max value along the embedding vector in the tensor for each question) and average similarity of question pairs (sum of values along the dimension in the tensor corresponding to the sequence of words in each question).
- Concatenate all the above features.
- Implement the following architecture:
  -> Dense(300, activation='relu')
  -> Dropout(0.5)
  -> BatchNormalization()
  -> Dense(100, activation='relu')
  -> Dropout(0.2)
  -> BatchNormalization()
  -> Dense(100, activation='relu')
  -> Dropout(0.2)
  -> BatchNormalization()
  -> Dense(50, activation='relu')
  -> Dropout(0.1)
  -> BatchNormalization()
  -> Dense(1, activation='sigmoid')

"The time distributed layer applies a dense layer to every temporal slice of an input. The input should be at least 3D, and the dimension of index one will be considered to be the temporal dimension. Consider a batch of 32 samples, where each sample is a sequence of 10 vectors of 16 dimensions. The batch input shape of the layer is then (32, 10, 16), and the input shape, not including the dimension of the sample, is (10, 16)." [3]

For the LSTM network:

- Add embedding layers for each question.
- Add LSTM layers for both questions, because it is a twin network, both inputs (questions) share the same LSTM.
- Combine the LSTM layers output using the Manhattan distance metric and concatenate with Jaccard and cosine similarity features.
- Implement the same architecture from here as for the time distributed network (see above).

The training-validation curves for the time distributed network are shown in Fig. 2. The calculation time on google colab was approximately 30 min.

The training-validation curves for the LSTM network are shown in Fig. 3. The calculation time on google colab was approximately 6 hours.

For the LSTM network a slightly better validation AUC was achieved at a cost of significantly increased computation time as well as less consistent and unstable loss decrease from epoch to epoch.

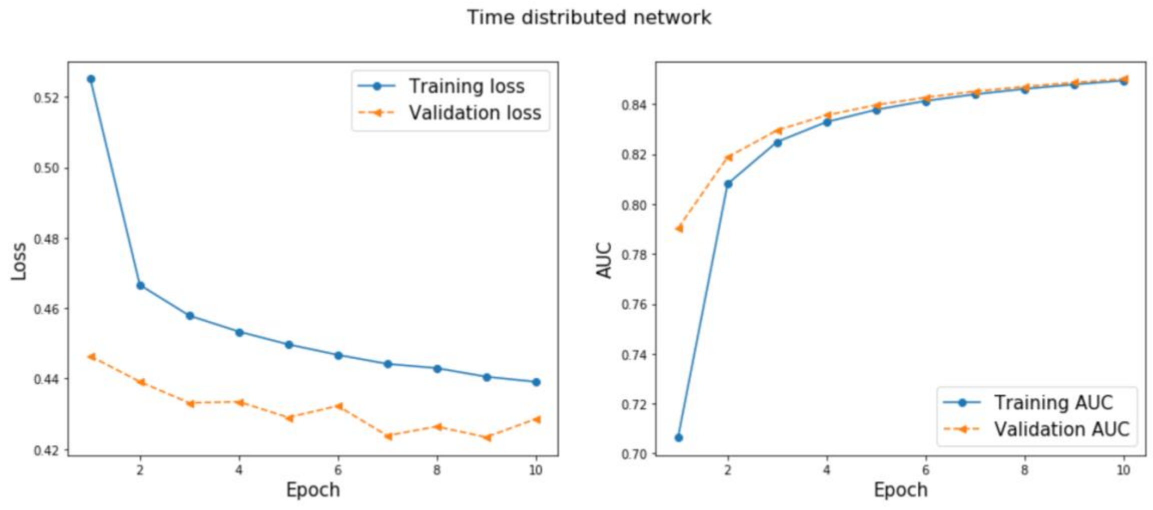

Fig. 2 Training-validation curves for the time distributed network.

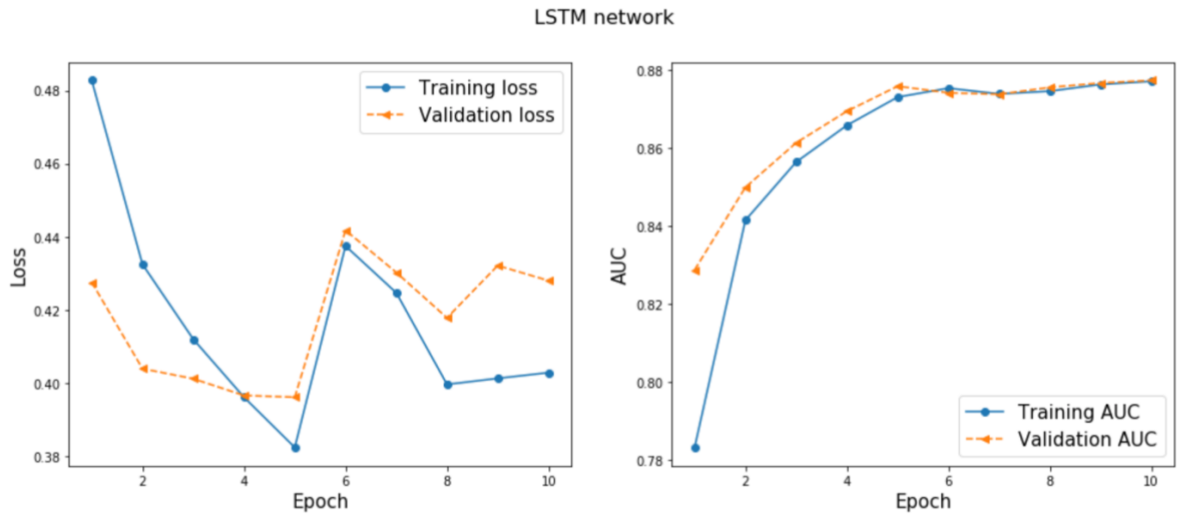

Fig. 3 Training-validation curves for the LSTM network.

The comparison of the networks' performance on the test dataset is shown in Table 1 and Table 2.

time distr.	precision	recall
not duplicate	0.82	0.84
duplicate	0.72	0.68
max valid. AUC		0.85

LSTM	precision	recall
not duplicate	0.81	0.87
duplicate	0.75	0.65
max valid. AUC		0.88

Table 1. Time distributed network.   Table 2. LSTM network.

Precision can be interpreted as: if I take a question pair predicted as duplicate, what is the probability that it is indeed a duplicate question pair.

Conclusions

- The twin neural networks with time distributed layers and LSTM layers were applied on the Quora question pairs dataset.
- Jaccard, cosine and Manhattan similarity metrics explored.
- Training the LSTM network was more computationally expensive, compared to the time distributed network, and resulted in a moderate increase of precision by 0.03 for duplicate samples (~6 hours for LSTM on google colab notebook with GPU for 10 epochs, compared to ~30 min for time distributed).

References

1. https://www.kaggle.com/c/quora-question-pairs/overview
2. https://www.kaggle.com/c/quora-question-pairs/data
3. https://keras.io/layers/wrappers/