Djona Fegnem

Capstone Project

Machine Learning Engineer Nanodegree

November 27, 2017

Project: Identify Pairs of Sentences with the Same Intent in Online User Forums.

Definition

Project Overview

Two questions with different vocabulary and syntactic structure asked on an online forum can have the same answer. Detecting semantic similarity between them can be a challenging task for traditional machine learning algorithms, which rely on a subject-mater expert to carefully extract useful features from data prior training. The advance of neural network[1] technology has provided methods to automatically uncover meaning in words by mapping them into high-dimensional vectors. This ability pairs with a recent computer vision technology called Convolutional Neural Network (CNN) has proven to be successful in detecting semantic similarity between pairs of sentences[2].

In this project, we will apply the same techniques on Quora dataset[3]. We will use as our baseline model an algorithm called eXtreme Gradient Boosting[4] (XGBoost) ,

[1] https://arxiv.org/abs/1301.3781

[2] https://www.aclweb.org/anthology/K15-1013

[3] https://www.kaggle.com/c/quora-question-pairs/data

trained on the same data after manually extracting useful features. Our experiment shows that the former method outperforms the latter.

Problem Statement

The goal is to create a model capable of identifying pairs of questions with the same answers. The tasks involved are the following:

1. Download and preprocess the Quora dataset.
2. Train a CNN classifier with the data.

The final model can be used to improve the user experience in online forum websites by quickly providing answers to previously asked and answered questions.

Metrics

For this project, we will use the f1 score as evaluation metric.

$$f1_{score} = 2 * \frac{precision * recall}{precision + recall}$$

$$precision = \frac{true\_positive}{true\_positive + false\_positive}$$

$$recall = \frac{true\_positve}{true\_positive + false\_negative}.$$

- true_positive: the number of similar pairs of sentences our model has successfully identified.

[4] https://xgboost.readthedocs.io/en/latest/

- false_positive: the number of similar pairs our model has identified as nonsimilar.
- false_negative: the number of nonsimilar pairs our model has identified as similar.
- The precision measures the proportion of positive instances correctly classified out of all the one returned by our model during evaluation.
- The recall measures the proportion of positive instances correctly classified out of all the positive instances returned by our model during our evaluation.

The f1 score is a single metric that captures both precision and recall.

Analysis

Data Exploration

Our data has 404,290 entries and 6 columns. The columns hold the following data:

1. Id: a unique integer value that represents the id of each entry.
2. qid1: a unique integer value that represents the id of the question 1 text.
3. qid2: a unique integer value that represents the id of the question 2 text.
4. question1: a text that represents the first question.
5. question2: a text that represents the second question.
6. is_duplicate: a value of 1 or 0 that indicates either similar or nonsimilar.

We have done the following during the preprocessing step:

1. Checked and removed missing entries[5].
2. Checked for duplicate questions and duplicate question pairs[6].

[5] section ‘check and remove missing entries’ in data_preprocessing.ipynb.

[6] section ‘Check for duplicates’ in data_preprocessing.ipynb

3. Removed all non-alphanumeric characters in sentences[7].
4. Removed the unnecessary columns: id, qid1, qid2.

At the end of our data preprocessing step, our final dataset has 404,281 entries and 3 columns, namely question1, question2, and is_duplicate.

Exploratory Visualization

Figure 1 and Figure 2 show some useful visualizations. *Figure 2* shows that we have more non-duplicate entries than we have duplicates. The non-duplicates account for roughly 63%, and the duplicates account for roughly 37% of our entire data. Furthermore, *Figure 1* suggests that sentences length are right skewed with values centered between 0 and 30 for question 1 and between 0 and 50 for question 2.

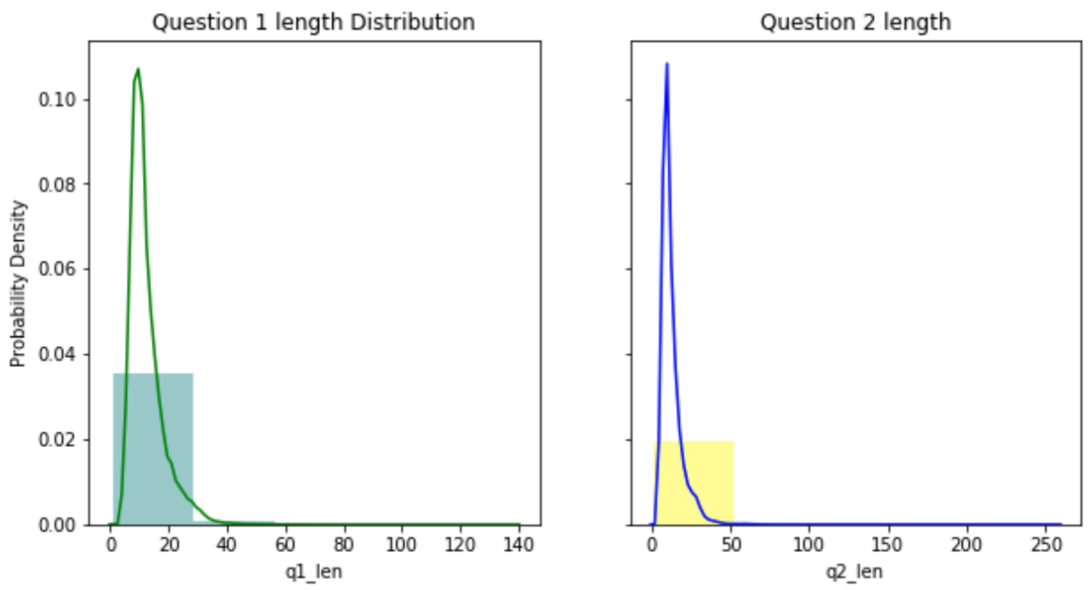


Figure 1 Distribution of sentences length

[7] section ‘Text Analysis’ in data_preprocessing.ipynb.

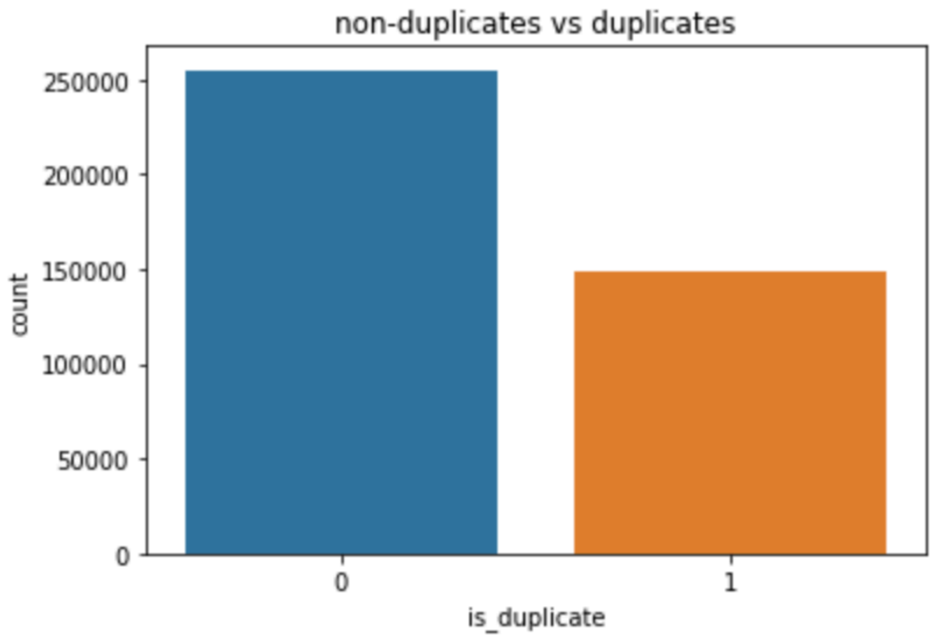


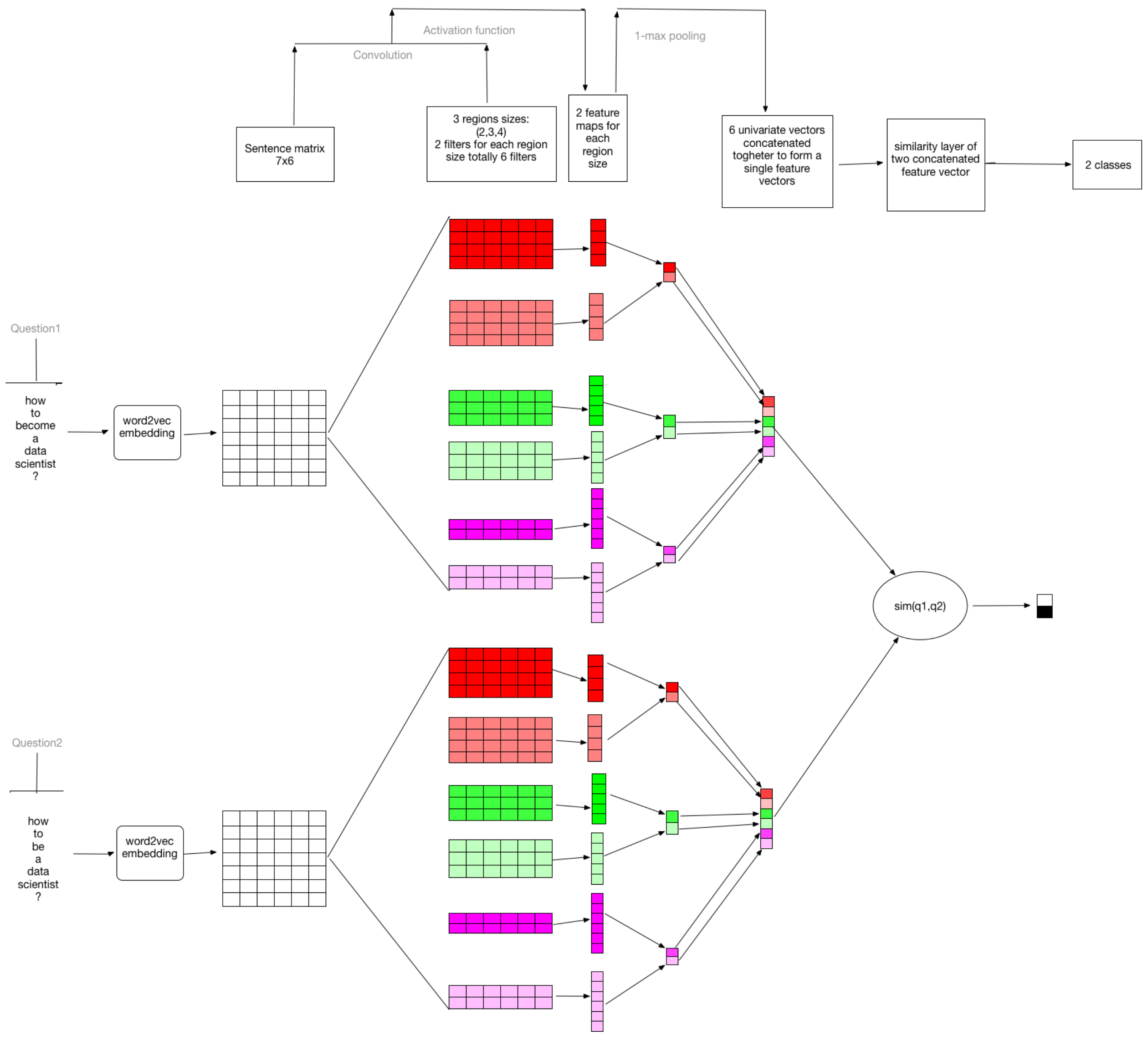
Figure 2 Number of duplicate pairs (1) and non-duplicate pairs (0)

Algorithms and Techniques

Our classifier is a Convolutional Neural Network (CNN). The training involves the following steps:

1. A tokenized pair of text is fed into an in-domain skip-gram neural network model. The model transforms each word into a high-dimensional vector.
2. The convolutional layer is used to construct two vector representations of each sentence.
3. The CNN computes the cosine similarity score between the two vectors. The cosine similarity score is a real value between 0 and 1. Scores close to 0

indicate that the pair is non-similar and scores close to 1 indicates that the pair is similar. We will use a threshold value during testing to discriminate between the two classes.



Activation function
Convolution
1-max pooling
Sentence matrix 7x6
3 regions sizes: (2,3,4) 2 filters for each region size totally 6 filters
2 feature maps for each region size
6 univariate vectors concatenated togheter to form a single feature vectors
similarity layer of two concatenated feature vector
2 classes
Question1
how to become a data scientist ?
word2vec embedding
sim(q1,q2)
Question2
how to be a data scientist ?
word2vec embedding

Figure 3 Network archittecture

Benchmark

As a baseline training, we used the XGBoost algorithm. We started by performing some feature engineering on the dataset. The goal of the feature engineering step is to extract useful features in our text data.

1. **Data preprocessing**[8]

   We have preprocessed the data by removing stop-words and punctuations.

2. **Feature engineering**[9]

   The following features have been extracted:

   - The cosine similarity score: this similarity score is calculated by using a vector representation of each text. This representation is calculated using a latent semantic structure model[10] computed by using singular vector decomposition applied on the term-document matrix with term-frequency inverse document frequency.
   - The number of tokens in each sentence.
   - The number of unique tokens in each sentence.
   - The number of shared tokens between the pair.

[8] Section ‘Remove punctuations and stopwords’ in data_preprocessing.ipynb

[9] Section ‘Feature engineering’ in data_preprocessing.ipynb

[10] http://lsa.colorado.edu/papers/JASIS.lsi.90.pdf

3. Visualizations

Figure 4 shows the distribution of our features. The questions length and questions vocabulary length are all right skewed. The ‘share_voc_len’ distribution is almost normal, and the similarity score distribution is binomial. *Figure 5* shows a strong positive relationship between the similarity score and the shared vocabulary length. *Figure 4* shows that the similarity score and the shared vocabulary length are the most important features in our dataset.

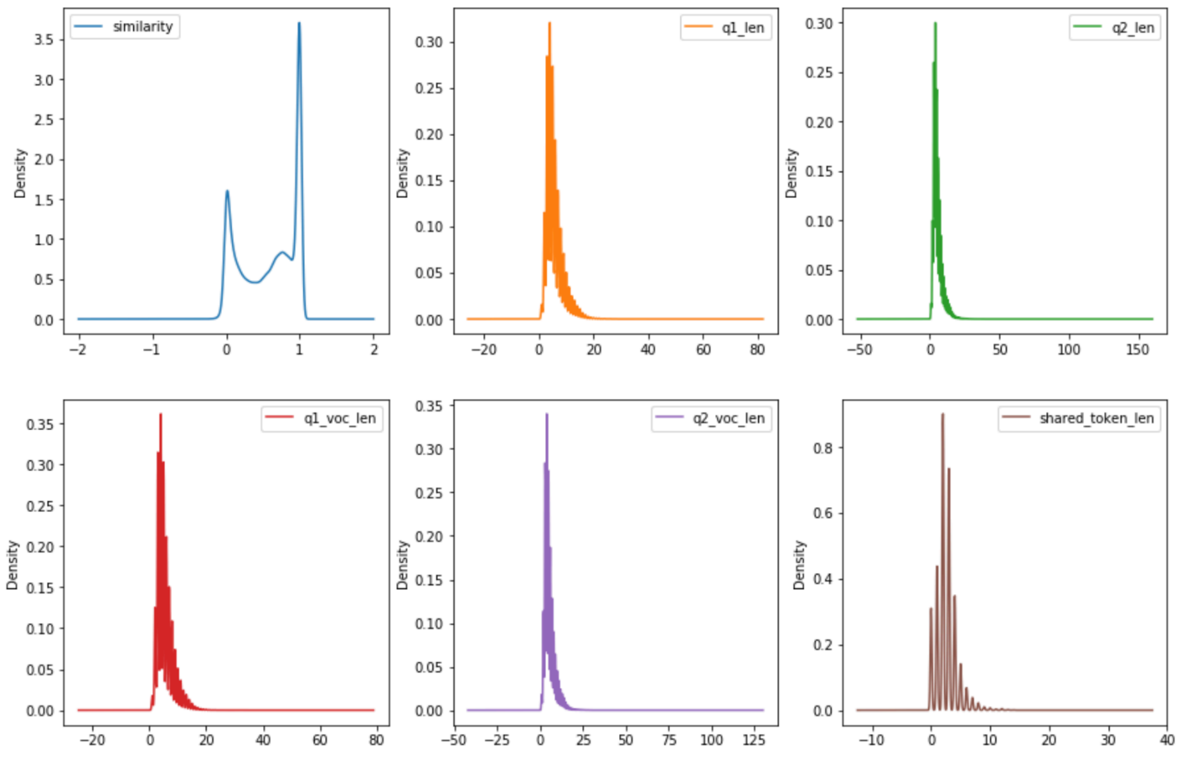


Figure 5 Features distribution

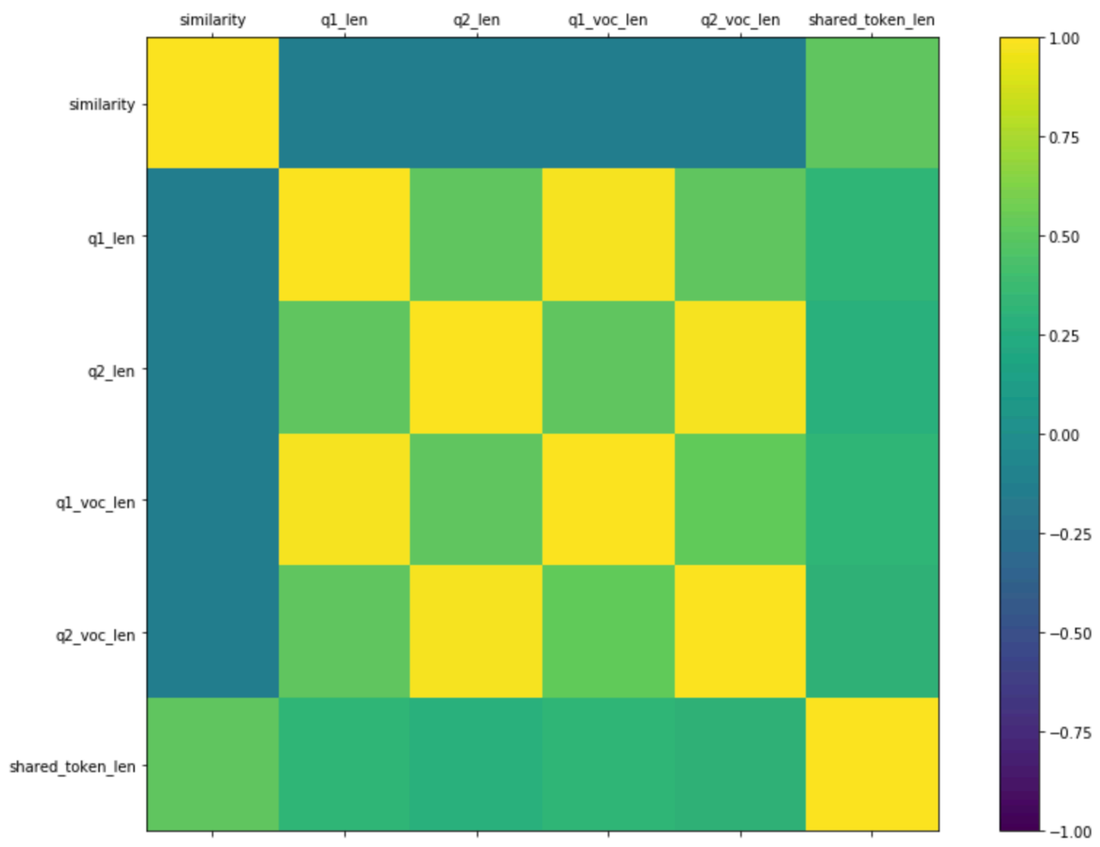


Figure 6 Correlation matrix

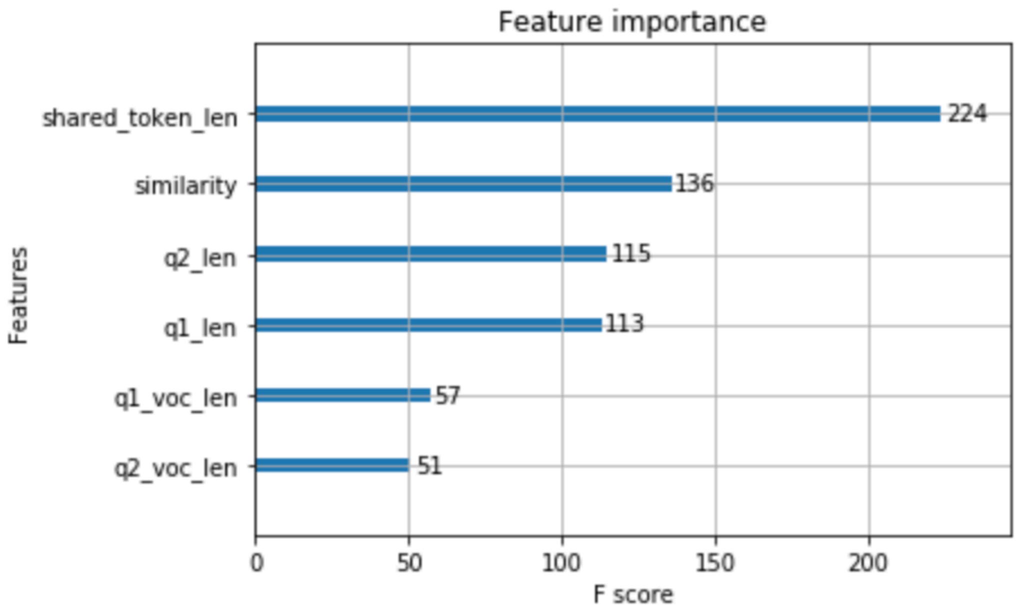


Figure 7 Feature importance

4. Split data[11]

After the feature engineering step, the dataset has been split into training[12] (90%, 3 53,625 instances) and test[13] sets roughly (10%, 40403 instances).

5. Training and Cross-validation

The benchmark model has been trained with the preprocessed dataset and the following parameters has been tuned:

- n_estimators: number of boosted trees to fit.
- learning_rate: learning rate.

[11] Section 'Split data for xgboost' in data_preprocessing.ipynb

[12] quora_xgb_train.pickle in data folder of the project

[13] quora_xgb_test.pickle in data folder of the project

- max_depth: maximum tree depth for base learners.

We used as objective function the logistic loss. We used the best hyper-parameters to fit our final model.

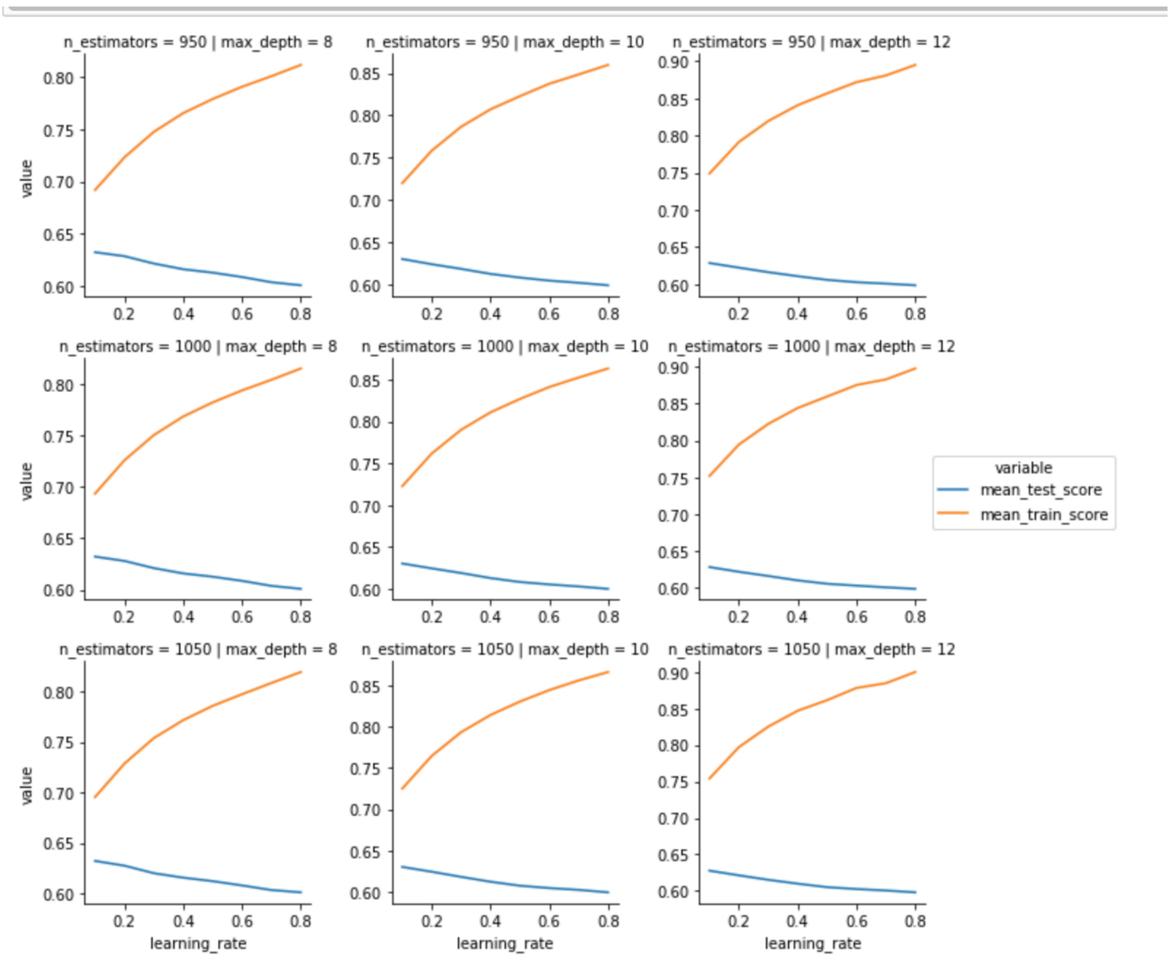


Figure 8 Hyperparameter cross-validation

6. Evaluation[14]

We have evaluated our final model on the test data. The final fbeta score we have obtained was 0.645.

[14] section 'final xgboost model' in data_preprocessing.ipynb

Methodology

Data preprocessing

To make our sentences the same length, we have padded[15] sentences less than 40 tokens with empty string, and we have only taken the first 40 tokens for sentences greater than 40 tokens. Furthermore to make our data suitable for CNN, we have created the vector representation of each word in sentences. More specifically, we have performed the followings on our data[16]:

1. Built an in-domain skip-gram neural network model[17] using our entire corpus text. We used the Gensim[18] - natural language processing tool- to build our skip-gram neural network model.
2. Used the model built earlier to get the word vector representation.

At the end of this preprocessing step, each sentence is replaced by a Numpy ndarray of real number with a shape equal of (sentence length, embedding length).

In other for each class in our data to have the same number of instances[19], we have taken all positive instances, and we have randomly selected the same number of

[15] Section 'Pad sentences' in data_preprocessing.ipynb

[16] Section 'word representations' in data_preprocessing.ipynb

[17] The code is in w2vec_model.py in the models folder.

[18] https://radimrehurek.com/gensim/

[19] Section 'split CNN data' in data_preprocessing.ipynb

instances within the negative class. The final dataset have 298,526 instances and 3 columns. To evaluate our model on unseen data, we have further divided our dataset into a training set[20] (90%, 268,673 instances) and a test set[21] (10%, 29,853 instances).

Convolutional Neural Network Definition

Once we have computed the vector representations of each word, our data is ready for CNN. The model architecture implementation[22] is described in as follow:

1. Input Layer: this layer has the pre-processed data.
2. 2D Convolution Layer: the goal of this layer is to capture spatial relationship among word using filters. We have applied different filters size. We have applied filter of size [2, 3, 4, 5, 6, 7, 8, 9, 10] to capture spatial relationship among - 2, 3, 4, 5, 6, 7, 8, 9, 10 - words that follow each other.
3. 2D max pooling layer: this layer reduces the size of the previous output by performing max pooling 2D operation on each previous output filter. The max pooling filter has size (sentence_len – filter_size + 1, 1).

[20] quora_cnn_train.pickle in data folder of the project

[21] quora_cnn_test.pickle in data folder of the project

[22] cnn_model.py

4. Merge Layer – Concat operation: this layer concatenate all the output filters from the previous layer for each sentence.
5. Flatten Layer: the output of the previous layer is then further flattened into a big vector representing the vector representation of each sentence.
6. Merge Layer – Cosine similarity operation: the vector of each sentence is then combined in this layer using a normalized dot product operation. The normalized dot produce measures the proximity of two vectors. The output of this layer is a real number between 0 and 1.

Training Procedure

Our network is trained by minimizing the mean squared error over the training set. Given a question pair *(q1, q2)*, the network with parameter set *W* computes a similarity score *s(q1, q2)*. Let *y(q1, q2)* be the correct label of the pair, where its possible values are 1(equivalent questions) or 0 (not equivalent questions). We used Adam[23] optimization algorithm with mini-batch to minimize the mean squared error with respect to *W* over the mini-batch. We used a batch size of 256 and a number of epochs of 100. The epoch is the number of time all the training set will be used to update our weights. At each epoch, our entire dataset is divided into mini-batches of equal size (256). At each pass, one mini-batch is randomly selected. The forward pass is computed using the mean squared error function loss.

[23] http://cs231n.github.io/neural-networks-3/

Then the backward pass computes the gradient using the back propagation algorithm. Finally, the Adam optimization algorithm is used to update our weights. Our network has been trained with Keras[24] and Tensorflow[25] as the backend.

Refinement

During training, we tried multiple filters size. We started with 32 filters of heights [2, 3, 4, 5]. Our evaluation score was around 0.7. By increasing the number of filters (512) and the number of filters height [2, 3, 4, 5, 6, 7, 8, 9, 10] we were able to improve our metrics. Furthermore, we have also applied the following: regularization technique[26]:

- Dropout regularization rate, with value of 0.3.
- Max-norm with value of 4.

[24] https://keras.io/

[25] https://www.tensorflow.org/

[26] http://jmlr.org/papers/v15/srivastava14a.html

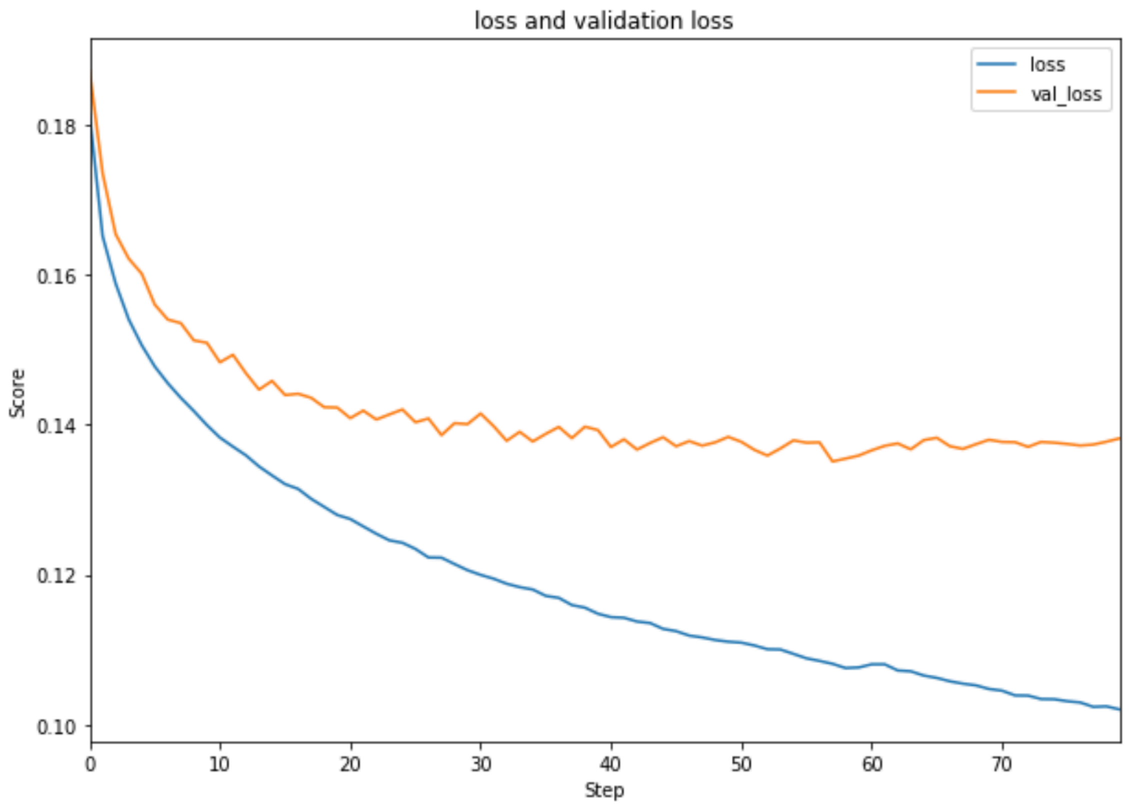


Figure 9 validation loss vs training loss

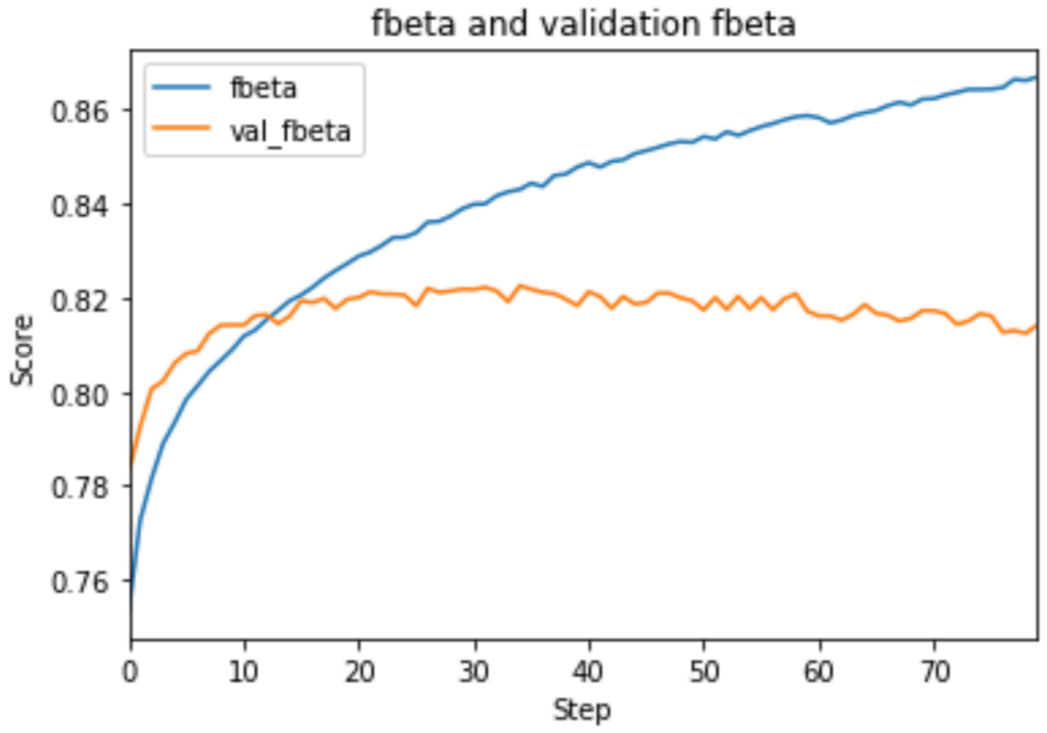


Figure 10 fbeta score

Results

Model Evaluation

Our model has been trained with a fixed threshold of k= 0.5 (threshold_shift = 0). A threshold of 0.5 means that predictions less than the threshold belong to the negative class, and predictions greater or equal than the threshold belong to the positive class. To choose the best threshold, we have evaluated our model on test data with different threshold value. The final threshold 0.7 (threshold_shift= -0.2) is the one that maximize the overall fbeta score of 0.836.

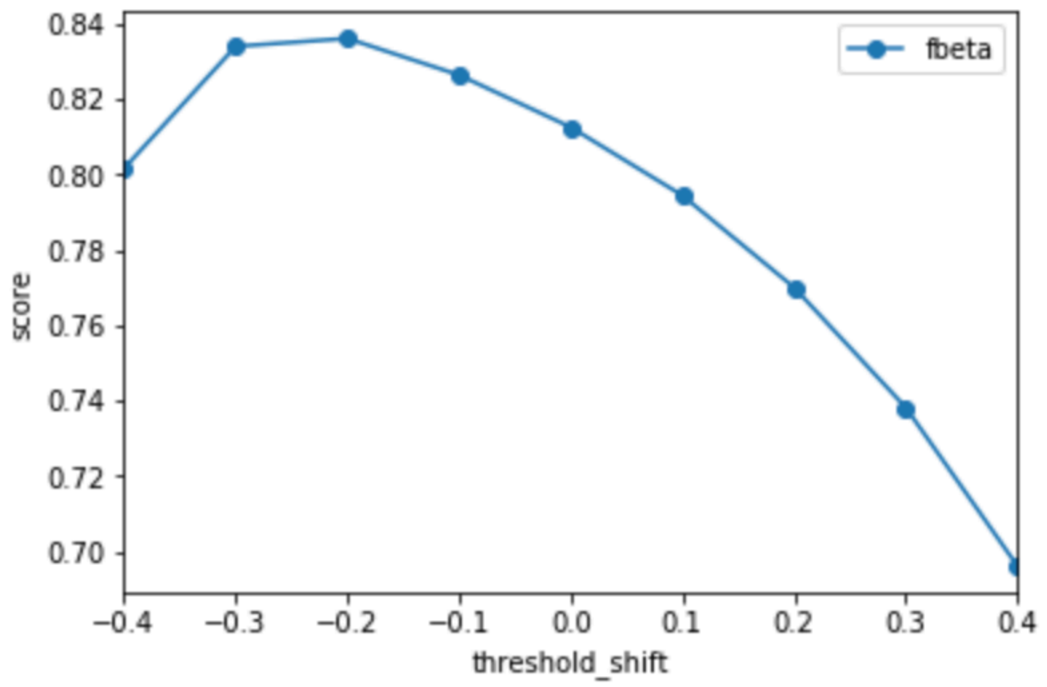


Figure 11 different threshold_shift

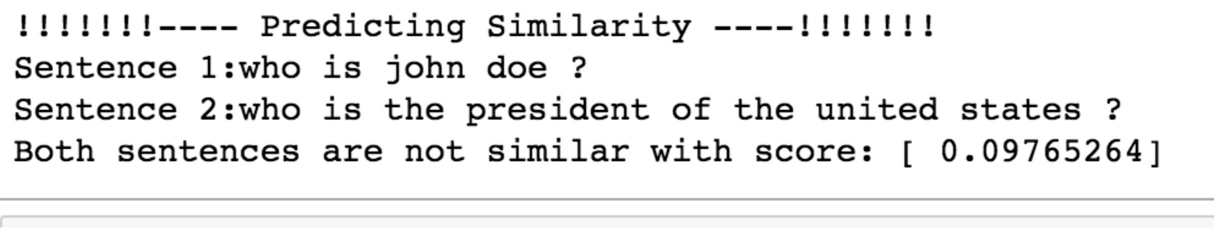
Justification

The CNN model performed better than the benchmark. This is due to the way both models are trained. The XGBoost model needs a human expert to extract useful features prior training, and the algorithm depends on his/her ability to identify useful features. The features we computed rely on individual tokens, they do not capture the relationship among tokens in the sentence. Additionally, we have attempted to extract domain knowledge of each sentence using an LSA model and use that knowledge to compute their similarity score in the vector space. LSA cluster documents (sentence) with the same topic close to each other in the space. It does not capture relationship among words within a sentence. Thus, this can explain the low fbeta score.

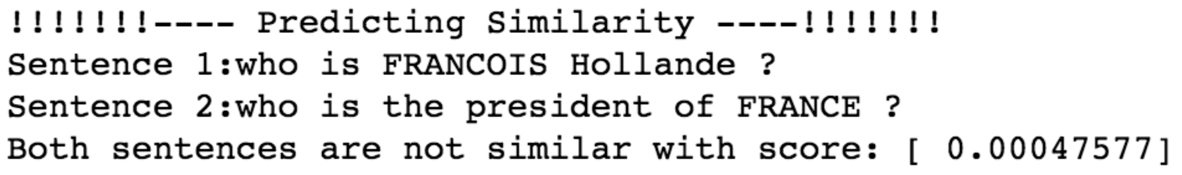
On the other hand, the CNN model extracts features automatically. The CNN capture local and spatial relationships among word within sentences and use that knowledge to compute their similarity. Everything is done automatically, we don't need any expert in other get good results. We got some good results within minutes of training. We just need to know how to best tune hyper-parameters.

Conclusion

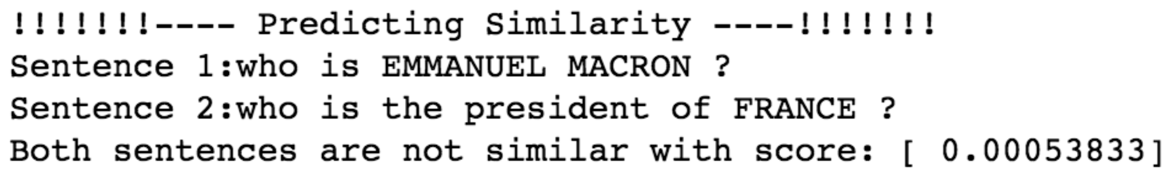
Free form visualization



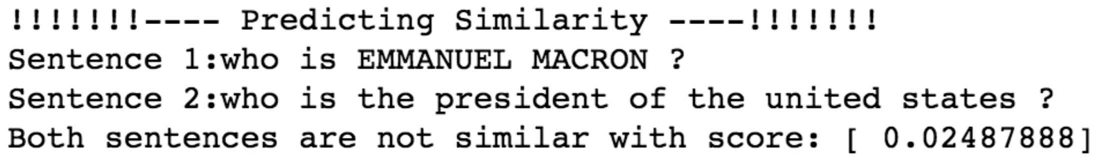
```
!!!!!!!---- Predicting Similarity ----!!!!!!!
Sentence 1:who is john doe ?
Sentence 2:who is the president of the united states ?
Both sentences are not similar with score: [ 0.09765264]
```



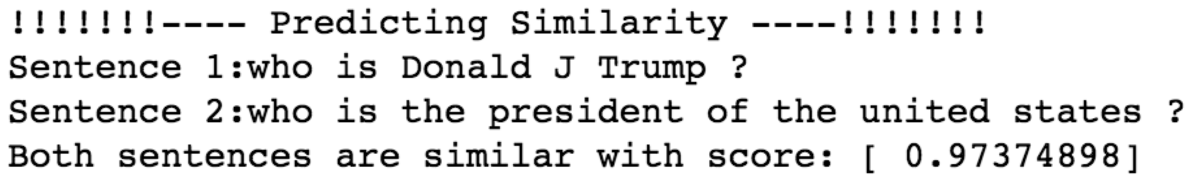
```
!!!!!!!---- Predicting Similarity ----!!!!!!!
Sentence 1:who is FRANCOIS Hollande ?
Sentence 2:who is the president of FRANCE ?
Both sentences are not similar with score: [ 0.00047577]
```



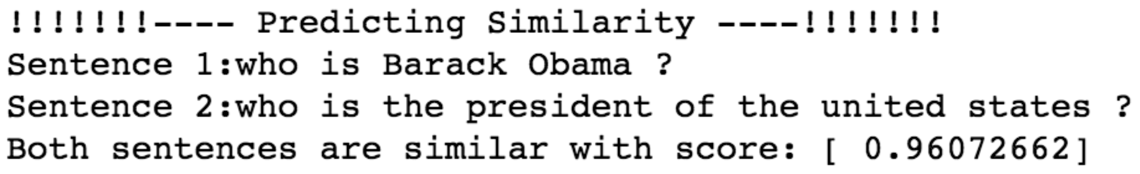
```
!!!!!!!---- Predicting Similarity ----!!!!!!!
Sentence 1:who is EMMANUEL MACRON ?
Sentence 2:who is the president of FRANCE ?
Both sentences are not similar with score: [ 0.00053833]
```



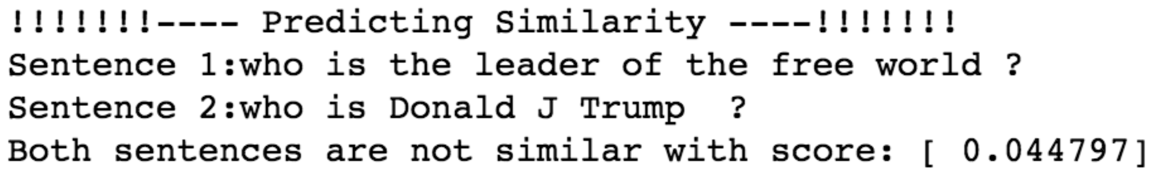
```
!!!!!!!---- Predicting Similarity ----!!!!!!!
Sentence 1:who is EMMANUEL MACRON ?
Sentence 2:who is the president of the united states ?
Both sentences are not similar with score: [ 0.02487888]
```



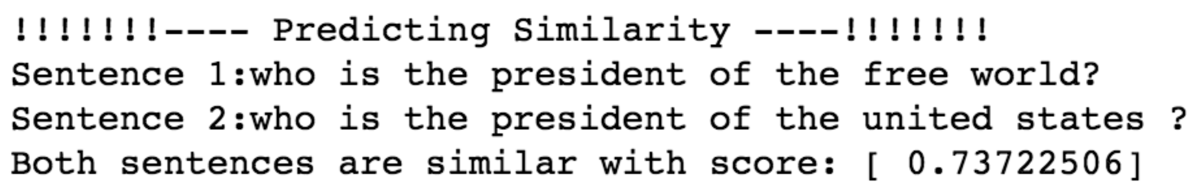
```
!!!!!!!---- Predicting Similarity ----!!!!!!!
Sentence 1:who is Donald J Trump ?
Sentence 2:who is the president of the united states ?
Both sentences are similar with score: [ 0.97374898]
```



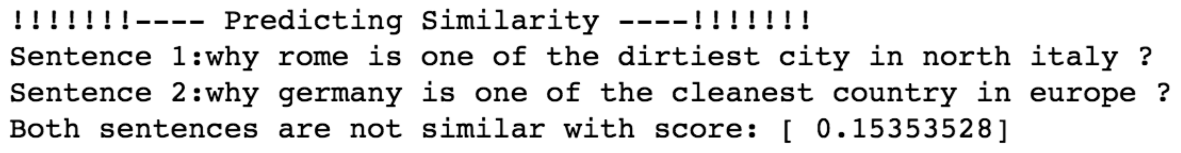
```
!!!!!!!---- Predicting Similarity ----!!!!!!!
Sentence 1:who is Barack Obama ?
Sentence 2:who is the president of the united states ?
Both sentences are similar with score: [ 0.96072662]
```



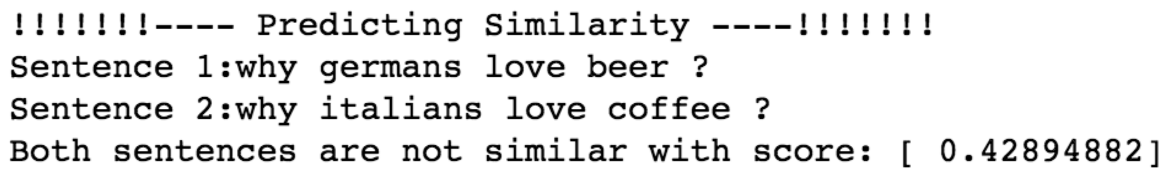
```
!!!!!!!---- Predicting Similarity ----!!!!!!!
Sentence 1:who is the leader of the free world ?
Sentence 2:who is Donald J Trump  ?
Both sentences are not similar with score: [ 0.044797]
```



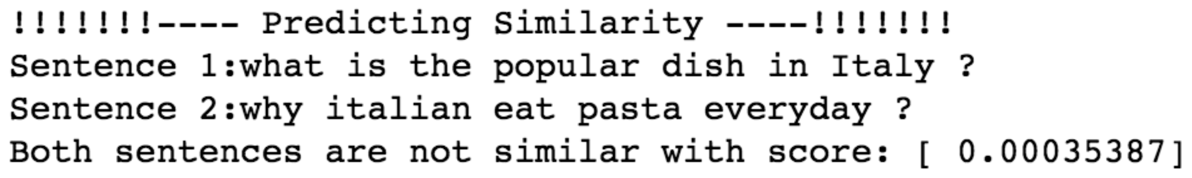
```
!!!!!!!---- Predicting Similarity ----!!!!!!!
Sentence 1:who is the president of the free world?
Sentence 2:who is the president of the united states ?
Both sentences are similar with score: [ 0.73722506]
```



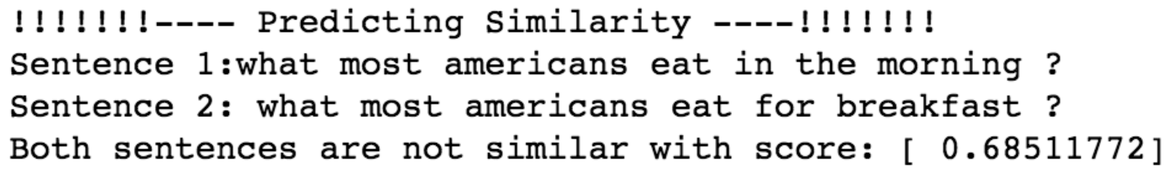
```
!!!!!!!---- Predicting Similarity ----!!!!!!!
Sentence 1:why rome is one of the dirtiest city in north italy ?
Sentence 2:why germany is one of the cleanest country in europe ?
Both sentences are not similar with score: [ 0.15353528]
```



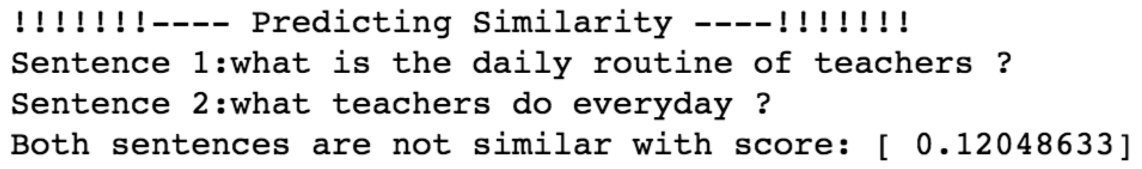
```
!!!!!!!---- Predicting Similarity ----!!!!!!!
Sentence 1:why germans love beer ?
Sentence 2:why italians love coffee ?
Both sentences are not similar with score: [ 0.42894882]
```



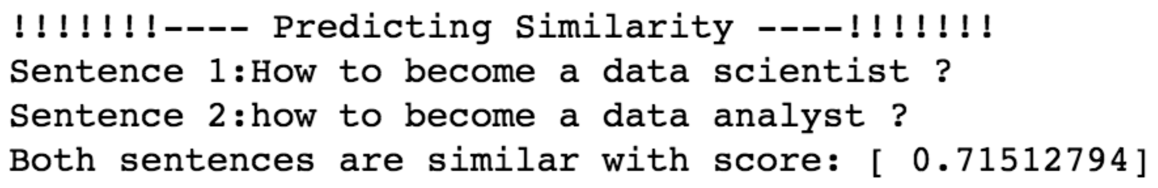
```
!!!!!!!---- Predicting Similarity ----!!!!!!!
Sentence 1:what is the popular dish in Italy ?
Sentence 2:why italian eat pasta everyday ?
Both sentences are not similar with score: [ 0.00035387]
```



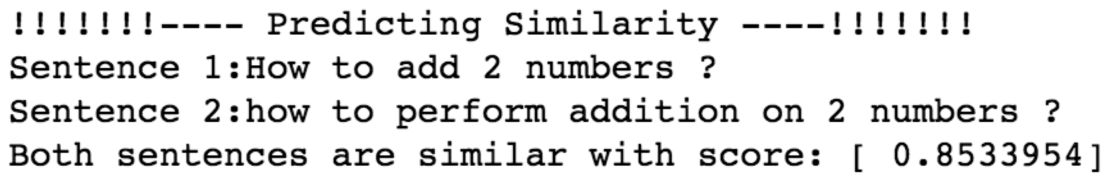
```
!!!!!!!---- Predicting Similarity ----!!!!!!!
Sentence 1:what most americans eat in the morning ?
Sentence 2: what most americans eat for breakfast ?
Both sentences are not similar with score: [ 0.68511772]
```



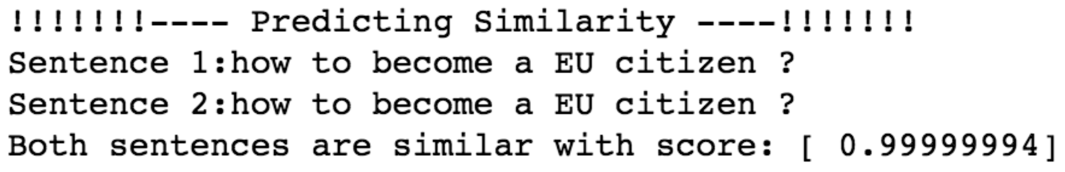
```
!!!!!!!---- Predicting Similarity ----!!!!!!!
Sentence 1:what is the daily routine of teachers ?
Sentence 2:what teachers do everyday ?
Both sentences are not similar with score: [ 0.12048633]
```



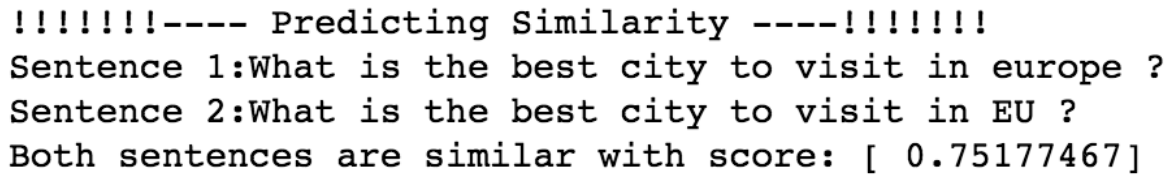
```
!!!!!!!---- Predicting Similarity ----!!!!!!!
Sentence 1:How to become a data scientist ?
Sentence 2:how to become a data analyst ?
Both sentences are similar with score: [ 0.71512794]
```



```
!!!!!!!---- Predicting Similarity ----!!!!!!!
Sentence 1:How to add 2 numbers ?
Sentence 2:how to perform addition on 2 numbers ?
Both sentences are similar with score: [ 0.8533954]
```



```
!!!!!!!---- Predicting Similarity ----!!!!!!!
Sentence 1:how to become a EU citizen ?
Sentence 2:how to become a EU citizen ?
Both sentences are similar with score: [ 0.99999994]
```



```
!!!!!!!---- Predicting Similarity ----!!!!!!!
Sentence 1:What is the best city to visit in europe ?
Sentence 2:What is the best city to visit in EU ?
Both sentences are similar with score: [ 0.75177467]
```

Reflection

Our problem was to identify question pairs with the same intent. We used as a baseline an XGBOOST model. We showed that traditional machine learning models require a domain expert to extract useful features. Our model performed better than the baseline. During the training process, I have encountered some challenges. The first one was during the datapreprocessing step, the challenge was to decide what tokens within sentences I should keep for training. Initially, I have removed English stop-words and punctuations in sentences, but by doing so my model performed poorly. As a result, I decided to keep all tokens. The model started having good results within few epochs (5). Another challenge that I faced was during hyper-parameter tuning. I tried L2 regularization, but there weren't any significant improvements. I ran it on about 10 epochs and compared the result. Using dropout and max-norm, I improved my model.

The final model can be used in a real-world application since misses won't affect negatively the user experience. The end user can help gather additional data to improve the model. This can be achieved by supplying the predicted similar question to the users when he/she asks questions. If the user agrees that the question he asked is similar to the one provided, we can provide him/her immediately with an answer.

Improvement

Besides collecting additional data from the end user, the model can be improved by tuning hyper-parameters as follow:

- Increase the embedding length of each word.
- Increase the number of filters
- Increase the number of epochs
- Try different values of drop-out, and max-norm
- Try different batch size

References

- Detecting Semantically Equivalent Questions in Online User Forums,

  Dasha Bogdanova.

- Efficient Estimation of Word Representations in Vector Space, Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean, Google Inc., Mountain View, CA.
- Indexing by Latent Semantic Analysis, Scott Deerwester, sept 1990.
- http://cs231n.github.io/