Semantic Similarity of Question Pairs

Student: Hui Gong hg2y@mtmail.mtsu.edu Instructor: Dr.Barbosa

1 Abstract

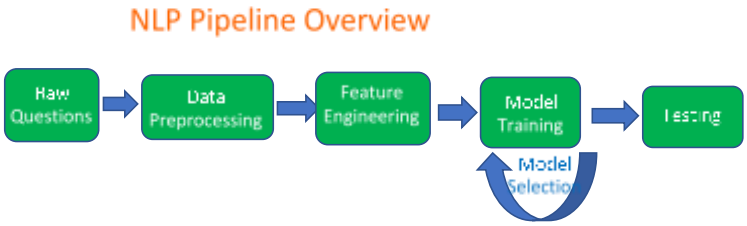
This project is for Class CSCI 6350 Natural Language Processing. Inspired by the Kaggle competition, "Quora Question Pairs – Can you identify question pairs that have the same intent?", this project is to apply the NLP techniques learned from the class to a real-world problem which helps me to have a better understanding of the techniques as well as get a taste of industrial NLP project. To determine if two questions have the same intention, a large corpus is used to train and test the word embedding methods, including tf-idf, word2vec, interrogative, etc., and the metrics of the confusion matrix is used to evaluate the performance of predictions. After a systematic search and testing, I selected tf-idf, word2vec, interrogative and combination of them all as features and combined with cosine similarity work on the Quora duplicate dataset. On my best model, I can get a result of 67% accuracy, 67.3% precision, 68.5% recall and 67.8% F1-score on the test set.

2 Introduction

Quora is a question-and-answer website which receives tons of questions every day. Questions with the same intent would cause seeker spend more time to get answers and providers have to answer similar questions multiple times. So solving this problem would be useful to help Quora organize and deduplicate their database.

3 Approach

The whole workflow is as shown:



3.1 Data

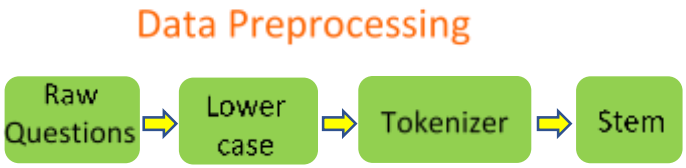
Use Quora duplicate questions dataset which contains 100,000 pairs of Questions which are already very similar to each other. I split the data into 80,000 pairs as training set and 20,000 pairs as test set.

id	qid1	qid2	question1	question2	is_duplicate
11	23	24	How do I read and find my YouTube comments?	How can I see all my Youtube comments?	1
27	55	56	Does society place too much importance on sports?	How do sports contribute to the society?	0
28	57	58	What is best way to make money online?	What is best way to ask for money online?	0
29	59	60	How should I prepare for CA final law?	How one should know that he/she completely prepare for CA final exam?	1

Table 1 Data set sample

3.2 Data preprocessing

The input is raw pairs of questions. Then use lowercase, tokenizer and stem to preprocessing the data. Here I do not remove the stop words which is very common for data preprocessing. Because questions usually are short, for example “where are you from?” After remove the stop words, nothing left. So I just keep them all.



3.3 Feature Engineering

3.3.1 TFIDF

Tf-idf, short for term frequency-inverse document frequency, is intended to reflect how important a word to a document in a corpus.

Term frequency (tf)

TF(t) = (Number of times term t appears in a document) / (Total number of terms in the document). Every document has its own term frequency.

$$tf_{i,j} = \frac{n_{i,j}}{\sum_k n_{i,j}}$$

Inverse Data Frequency (IDF)

IDF(t) = log_e(Total number of documents / Number of documents with term *w* in it).

$$idf(w) = \log(\frac{N}{df_t})$$

Tfidf is TF multiplied by IDF:

$$w_{i,j} = tf_{i,j} * idf(w)$$

3.3.2 Word2Vec

I convert unstructured text data into numeric feature vectors using word2vec.

Word2vec is a group of related models that are used to produce word embedding. It takes as its input a large corpus of text and produces a vector space, usually of hundreds dimensions, with each unique word in the corpus being assigned a corresponding vector in the space. Here I used genism pre-trained word vectors to initialize our word embedding. And choose 300-dimensional vectors for our problem.

I calculate the average of vector for a given question. First for every word in the question, if the word in the model vocabulary, get its vector. Then sum all the vector. At last, calculate the average.

3.3.3 Interrogative

If two questions have the same intent, it’s highly possible that they have the same interrogative. Otherwise, they are not. Here is the collection of interrogatives:

interrogative = ["who", "whose", "when", "where", "what", "which", “how”]

Like if a question is: “Where are you from?”, then it should return a vector of [0,0,0,1,0,0,0]

4 Model: Cosine Similarity

The size of data set as discussed in section 3.1 is 100,000, which will be time-consuming for complicated machine learning models like Neural Networks. To minimize the time and effort needed on model training, I need to choose a method which is easy to implement and fast to run. Cosine similarity is a widely used predictive method for large data set, which strikes a good balance between efficiency and accuracy.

Cosine similarity of a measure of similarity between two non-zero vectors of an inner product space that measures the cosine of the angle between them. Two vectors with the same orientation have a cosine similarity of 1. And two vectors oriented at 90 degree, the similarity is 0.

$$similarity = \cos(\theta) = \frac{A.B}{|A|\,|B|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

5 Experiment

This section details the major steps I discussed in the above section, and apply them to a data set discussed in Section 3.1. The goal is to find the optimal configuration of word embedding method and predictive model. As I discussed above, cosine similarity is the method of choice for this experiment.

Step 1. For a given pair of questions, get vector for each question under a feature

Step 2. Calculate the cosine similarity with the pair vector

Step 3. Set a list of thresholds from 0 to 1, every step as 0.05

Step 4. Using the similarity score to predict result at each threshold

Step 5. Repeat step 1 to 4 to run all the training set

Step 6. Calculate accuracy, precision, recall and F1-score to choose the best threshold

Step 7. Using trained model to predict the test set

5.1 Evaluation

Various metrics are available in both academic and industry in evaluating the performance of models. For this project, I use the commonly used confusion matrix.

Confusion matrix is a specific table layout that allows visualization of performance of an algorithm. Each row of the matrix represents the instances in a predicted class while each column is the instances in a true condition.

		True condition	
		positive	negative
Predicted	postive	true positive	false positive
	negative	false negative	true negative

The major indicators I focus on are precision, recall, accuracy, and F1-score. I first briefly discuss the four indicators.

Accuracy is the most intuitive performance measure and it is a ratio of correctly predicted observation to the total observations.

$$Accuary = \frac{TP + TN}{TP + FP + TN + FN}$$

Precision is about how precise/accurate model is out of predicted positive, how many of them are actual positive.

$$Precision = \frac{TP}{TP + FP}$$

Recall calculates how many of the actual positives the model capture.

$$Recall = \frac{TP}{TP + FN}$$

F1 score is the balance between precision and recall.

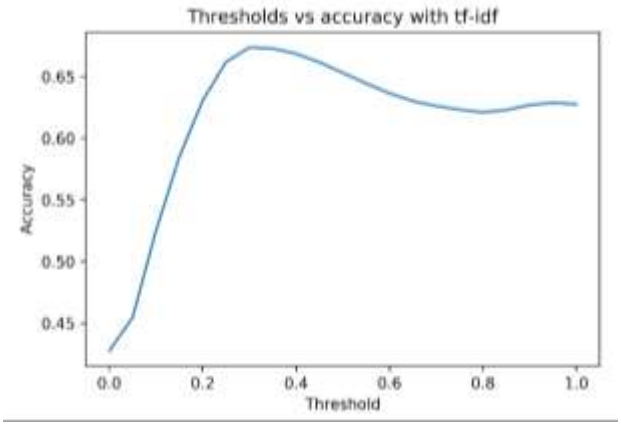
$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall}$$

The drawbacks of these metrics, such as treating false positive and false negative equally, are not the focus of this project. So, I will not have a further discussion here.

5.2 Training model

In the following section, the effectiveness of tf-idf, word2vector, interrogative and the combination of them as features are tested. Cosine similarity is chosen as the method to predict the labels of training set. Cosine similarity requires the threshold to determine if two questions have the same attention. And the optimal threshold is one of the key factors that determines the performance on test results that I will perform on test set later. To find the turning point of threshold, values from 0 to 1 with step size of 0.05 are examined. Below are the results of each embedding method. The tables show the accuracy, precision, recall and F1-score for each threshold, and the figures plot the tables which helps visualize the relationship between threshold and accuracy. For each embedding method, I'd like to find the turning point at which the accuracy is the highest where the threshold value is the smallest.

5.1.1 Tf-idf with cosine similarity

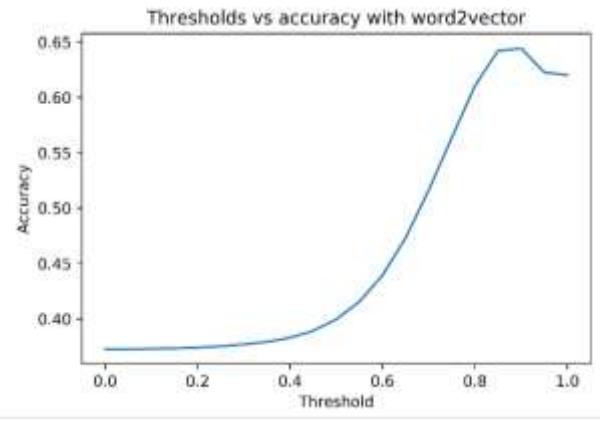


Threshold	Accuracy	Precision	Recall	F1-score
0.00	0.43	0.70	0.54	0.61
0.05	0.45	0.70	0.57	0.63
0.10	0.52	0.71	0.62	0.66
0.15	0.58	0.71	0.66	0.68
0.20	0.63	0.70	0.69	0.69
0.25	0.66	0.69	0.70	0.69
0.30	0.67	0.67	0.69	0.68
0.35	0.67	0.66	0.66	0.66
0.40	0.67	0.64	0.64	0.64
0.45	0.66	0.63	0.62	0.62
0.50	0.65	0.62	0.60	0.61
0.55	0.64	0.61	0.57	0.59
0.60	0.64	0.59	0.55	0.57
0.65	0.63	0.58	0.54	0.56
0.70	0.63	0.57	0.53	0.55
0.75	0.62	0.56	0.52	0.54
0.80	0.62	0.54	0.51	0.52
0.85	0.62	0.53	0.51	0.52
0.90	0.63	0.56	0.50	0.53
0.95	0.63	0.72	0.50	0.59
1.00	0.63	0.74	0.50	0.60

Table 1 Threshold with feature tfidf

For tf-idf with cosine similarity, from the figure and table above, I can see that when the threshold is 0.30, the accuracy reaches the maximum, which is about 0.67, and this threshold will be used as the parameter for our test set later. The same process is performance for word2vec, and interrogative and the combination. The results are shown in the following. (In order to save the space the following table's just show thresholds with step size 0.1 instead of 0.05)

5.2 Word2Vec with cosine similarity

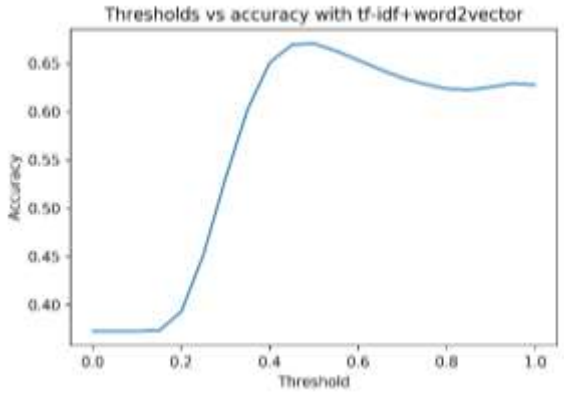


Thresh	Accura	Precisi	Recall	F1-scor
0	0.37	0.62	0.5	0.55
0.1	0.37	0.64	0.5	0.56
0.2	0.37	0.67	0.5	0.57
0.3	0.38	0.68	0.5	0.58
0.4	0.38	0.68	0.51	0.58
0.5	0.4	0.68	0.52	0.59
0.6	0.44	0.67	0.55	0.61
0.7	0.52	0.66	0.6	0.63
0.8	0.61	0.64	0.65	0.64
0.9	0.64	0.61	0.59	0.6
1	0.62	0.46	0.5	0.48

Table 2 Threshold with feature word2Vec

For word2vec with cosine similarity, from the figure and table above, I can see that when the threshold is 0.87, the accuracy reaches the maximum, which is about 0.64, and this threshold will be used as the parameter for our test set later.

5.3 Tf-idf + word2Vec with cosine similarity

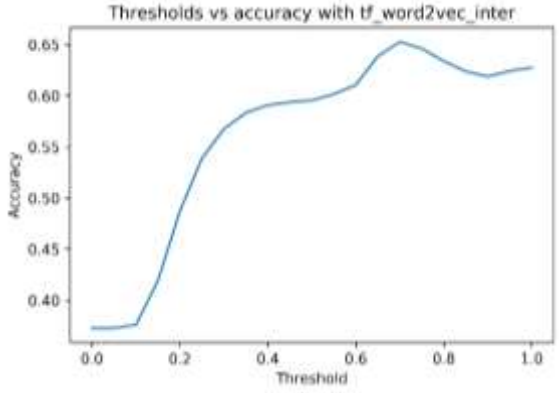


thresho	Accurac	Precisi	Recall	F1-scor
0.00	0.37	0.69	0.50	0.58
0.10	0.37	0.69	0.50	0.58
0.20	0.39	0.69	0.52	0.59
0.30	0.53	0.71	0.62	0.66
0.40	0.65	0.69	0.69	0.69
0.50	0.67	0.65	0.66	0.66
0.60	0.65	0.62	0.60	0.61
0.70	0.63	0.59	0.55	0.57
0.80	0.62	0.56	0.52	0.54
0.90	0.63	0.55	0.51	0.53
1.00	0.63	0.56	0.50	0.53

Table 3 Threshold tfidf + word2Vec

For the combination of tf-idf and word2vec with cosine similarity, from the figure and table above, I can see that when the threshold is 0.50, the accuracy reaches the maximum, which is about 0.67, and this threshold will be used as the parameter for our test set later.

5.4 Tf-idf + word2Vec + interrogative



Threshold	Accuracy	Precision	Recall	F1-score
0	0.37	0.69	0.5	0.58
0.1	0.38	0.69	0.5	0.58
0.2	0.49	0.67	0.58	0.62
0.3	0.57	0.64	0.62	0.63
0.4	0.59	0.62	0.62	0.62
0.5	0.6	0.61	0.61	0.61
0.6	0.61	0.61	0.61	0.61
0.7	0.65	0.62	0.62	0.62
0.8	0.63	0.59	0.55	0.57
0.9	0.62	0.53	0.51	0.52
1	0.63	0.31	0.5	0.39

Table 4 Threshold tfidf + word2Vec + interrogator

For Tf-idf + word2Vec + interrogative with cosine similarity, from the figure and table above, I can see that when the threshold is 0.70, the accuracy reaches the maximum, which is about 0.65, and this threshold will be used as the parameter for our test set later.

5.3 Test model

After the above training model, I get the threshold [0.30, 0.87,0.50 ,0.70] of cosine similarity score for feature tf-idf, word2Vec, tf-idf + word2Vec and tf-idf + word2Vec + interrogative.

For the 20,000 pairs of test questions, the result is shown as below.

Threshold	Accuracy	Precision	Recall	F1-score
0.300	0.671	0.673	0.685	0.678
0.870	0.649	0.626	0.627	0.626
0.500	0.672	0.656	0.663	0.659
0.700	0.652	0.625	0.621	0.623

Table 5 Test Result

From the result, I can get accuracy is almost higher than 0.65 for almost all the feature that I choose. Which shows our model performs good. Using Tf-idf as feature with the threshold at 0.30 can get the highest value on the accuracy, precision, recall and F1-score.

I also tried other features such as count the proportion of nouns/verbs in a sentence or using named entity recognition to compare the location in the questions. It did not improve the performance of the model or very time consuming.

5.3 Further development

I use the pre-trained word vectors for word embedding initialize. If based on word embedding trained on Quora data, I believe it would get a much better result. On the other hand, cosine similarity is a very effective algorithm but may not perform the best. In the further, I can other method such as word move distance or neural network.

6 Conclusion

In this project, I used different features combining with cosine similarity to work on the Quora duplicate dataset problem. When taking tf-idf as feature at the threshold set 0.30, I can get a result of accuracy 67%, precision 67.3%, recall 68.5% and F1-score 67.8%. I will try different algorithm to improve the performance in the further.

References

[1] https://www.kaggle.com/c/quora-question-pairs/overview
[2] Sharma L, Graesser L, Nangia N, Evci U. Natural Language Understanding with the Quora Question Pairs Dataset. arXiv preprint arXiv:1907.01041. 2019 Jul 1.
[3] Jonas Mueller and Aditya Thyagarajan. Siamese Recurrent Architectures for Learning Sentence Similarity. In AAAI, 2016.
[4] Daniel Jurafsky and James H. Martin. Speech and Language Processing, 3rd Edition DRAFT
[5] Steven Bird, Ewan Klein, and Edward Loper. Natural Language Processing with Python, O'Reilly Media, 2009
[6] https://towardsdatascience.com/natural-language-processing-feature-engineering-using-tf-idf-e8b9d00e7e76
[7] https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1174/reports/2761178.pdf
[8] https://en.wikipedia.org/wiki/Cosine_similarity
[9] https://en.wikipedia.org/wiki/Tf%E2%80%93idf
[10] https://medium.com/@shritamkumarmund.98/quora-question-pair-similarity-case-study-54f0d8c9b630