Quora Question Pairs

Yifan Liu, Haosen Cheng

Introduction

Searching online, how do we know if a question is asked before? Many people ask similar questions, which have the same intend causes time wastes. For example, “Where are you from?” and “Where do you come from?” are basically the same question in different forms. Our job is to develop natural language processing and machine learning system to classify whether a question pair is duplicated or not. This could be very useful, by the result provided by this classification, we can provide answers directly to questions have been asked before.

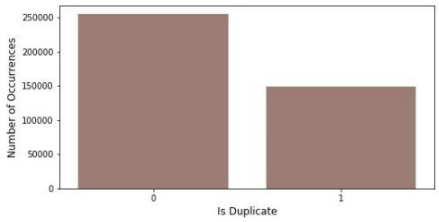
In general, Quora used a random forest model to classify question pairs, in our project, we use other techniques which will illustrate below, the baseline error rate is 0.5, after applied our techniques, we made a 20% improvement.

Data Preprocessing and Feature Engineering

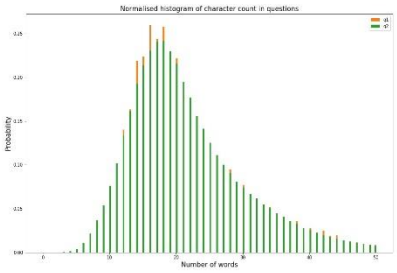
	id	qid1	qid2	question1	question2	is_duplicate
0	0	1	2	What is the step by step guide to invest in sh...	What is the step by step guide to invest in sh...	0
1	1	3	4	What is the story of Kohinoor (Koh-i-Noor) Dia...	What would happen if the Indian government sto...	0
2	2	5	6	How can I increase the speed of my internet co...	How can Internet speed be increased by hacking...	0
3	3	7	8	Why am I mentally very lonely? How can I solve...	Find the remainder when [math]23^{24}[/math] i...	0
4	4	9	10	Which one dissolve in water quikly sugar, salt...	Which fish would survive in salt water?	0

Looking at the training set, we have an id for each question pair and question id for each question in the pair, followed by our questions and a mark indicate if it is duplicate or not. What we need here is the two questions in the pair, but first let’s look at some statistics of the data.

First, we have a look at the number of question pairs marked as duplicate. As shown in the histogram on the right, 36% of the training set is marked duplicates over 404290 pairs in the training set.



The right-side figure shows the distribution of sentence length in both training set and test set. For each question pair, we use the total length, meaning that the length in the plot is the length of two sentences. This gives out a similar mean and variance.



We believe data cleaning is quite important. We changed every word in the data into lowercase. We filled all the NaN cells with an empty string. Furthermore, we changed some words in the dataset. For example, “I’ll to I will”, “having to have” and “ios to the operating system”. By doing the data cleaning, we improved the accuracy of our model by 3%. We believe this is not the best that data cleaning can brings. If we have more time, we will try other cleaning methods to see whether we can push our model to a higher level.

Feature Extraction

- Jaccard similarity

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}$$

We used Jaccard similarity as one of the baseline model. A and B in the formula are the length of sentence 1 and sentence 2. The Jaccard coefficient measures similarity between finite sample sets and is defined as the size of the intersection divided by the size of the union of the sample sets. [1]

- Cosine similarity

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

We used Cosine similarity as another baseline model. The cosine coefficient measures similarity between 2 vectors. It is represented using a dot product and magnitude. [2]

From two baseline models, we believe Jaccard algorithm performs better in our tasks. Therefore, we used Jaccard similarity as our base model and improved it with a combination of n-gram. We calculated Jaccard similarity by the length calculated by 2-gram and 3-gram.

- TF-IDF (term-frequency-inverse-document-frequency)

$$w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right)$$

Where $tf_{i,j}$ is the number of occurrence of word i in sentence j, df_i is the number of sentences contain word i and N is the total number of documents.

TF-IDF is a method for scoring a word to see if it is important or not, by taking average of this score in a sentence, we get a TF-IDF score for each sentence. This will be used later in the model.

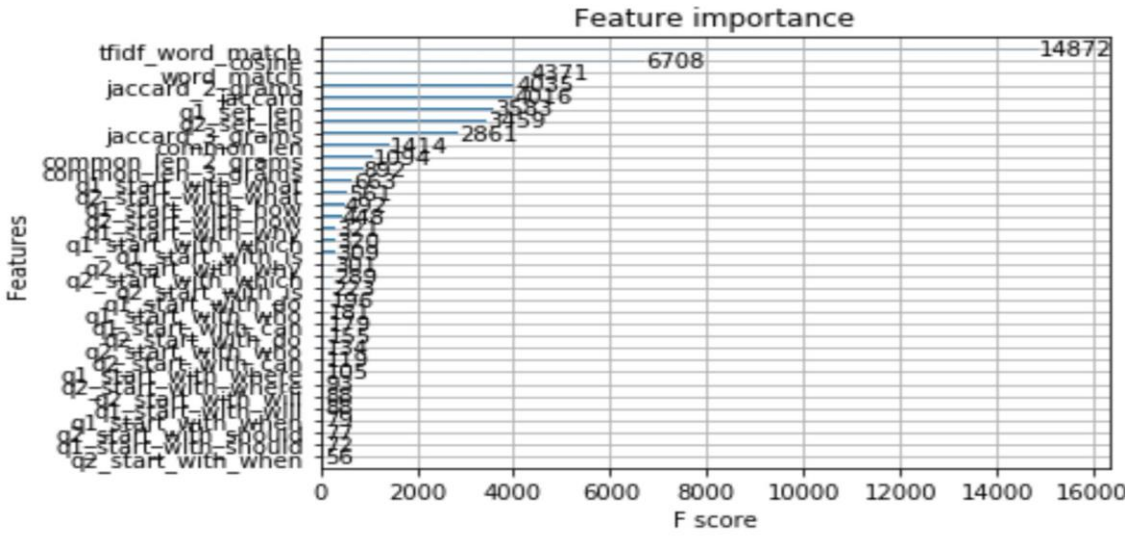
Model training

After we have all the features mentioned above, we can start model training. We choose to use XGBoost[3] as our model because it is light weight, easy to use and good performed. XGBoost is an optimized gradient boosting library for efficient, flexible and portable training. It is implemented under gradient boosting framework.

We use this to train our 35 features training set and get a 0.4 error rate. Initially using just, the similarities, we get a score at 6.5. Combining and adding features gives us a huge improvement. The plot below shows the importance of features we used. We can clearly see that the jaccard with 2-gram, TF-IDF and Cosine similarity are the most useful features.

Looking online, we find the best score on Kaggle is 0.11 using deep learning network.

Current score: 0.4



Conclusion & Further work

For now, we can compare similarity between two English questions. By changing feature extraction methods, we can extend this to all English sentences and by adding word alignment algorithms implemented in homework 4, we can further improve this to compare sentences among English, French and German.

Contribution

Yifan Liu: Modeling XGBoost, TF-IDF, training and testing

Haosen Cheng: Data cleaning, implement jaccard and cosine similarity, n-gram and combine files with Yifan

Zhuoli Xie and Xin Wei: Tester.

Reference

1. https://en.wikipedia.org/wiki/Jaccard_index
2. https://en.wikipedia.org/wiki/Cosine_similarity
3. https://xgboost.readthedocs.io/en/latest/
4. Wang, Z., Hamza, W., & Florian, R. (2017). Bilateral Multi-Perspective Matching for Natural Language Sentences. Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence. doi:10.24963/ijcai.2017/579
5. https://towardsdatascience.com/overview-of-text-similarity-metrics-3397c4601f50