Kaggle Competition: Quora Question Pairs
ENSC895 Course Project

Arlene Fu, 301256171
Professor: Ivan Bajic
Simon Fraser University
December 4th, 2017

1. Introduction

There are over 100 million people visiting Quora every month, it is quite possible that people ask similarly worded questions. Quora uses the random forest model to classify duplicate questions currently. The goal of this challenge is applying advanced techniques to evaluate whether the provided pairs of questions deliver the same meaning. Once successfully identify duplicate questions, users can easily find high quality answers without spending more time looking for best answer among multiple similar questions, and also less time for writers to answer the same questions multiple times.

2. Project Description

This is a Kaggle competition hold by Quora, it has already finished six months ago. The goal of this competition is encouraging competitors to develop a machine learning and natural language processing system to classify whether question pairs are duplicates or not.

In other words, this semantic question matching problem can be defined as follows: for question pair q1 and q2, train a deep learning model to predict the function:

$$f(q1, q2) \rightarrow 0 \; or \; 1$$

0 represents that q1 and q2 are not duplicate.

2.1. Dataset

The dataset for this competition is provided on Kaggle website [1]. The website provides a training dataset, which contains more than 400,000 pairs of questions. Test dataset is also available, which has over 2.3 million pairs of questions. The size of train and test sets are 64MB and 314MB respectively. Both of two sets are in .csv format. Training set contains the following data field:

id	qid1	qid2	question1	question2	Is_duplicate
0	1	2	What is the step by step guide to invest in share market in india?	What is the step by step guide to invest in share market?	0
1	3	4	…	…	…
…	…	…	…	…	…

'id' is just a simple row ID, 'qid{1,2}' are the unique ID of each question in the pair. The full text of each question is given in 'question1' and 'question2'. 'is_duplicate' is the target value. The test set consists of 'test_id', 'question1' and 'question2'.

test_id	question1	question2
…	…	…

2.2. Evaluation

Log Loss (binary cross-entropy loss) is used as evaluation metric.

$$LogLoss = -\frac{1}{N}\sum_{i=1}^{N}[y_i log p_i + (1 - y_i)\log(1 - p_i)]$$

2.3. Data Analysis

According to competition website, training questions are genuine examples come from Quora, but most question pairs in the test set are computer-generated. For the rows which do not come from Quora, are not counted in the scoring. This means the true number of rows that are scored in test set is less than 2.3 million.

Therefore, I first tokenize all the sentence in both training and test set and build two self-defined dictionaries to store all the words. The dictionary records frequency of occurrence for each word. It is easily to get the words which appear in test set but not in training set. Using these words, I find there are lots of meaningless sentences. For example:

Can you weren share your diet plan?

What food fibre?

3. Implementation

The work I implement is mainly based on Ahmet Erdem's code [2]

3.1. Data pre-processing

Before building deep learning model, I perform some data pre-processing. First, tokenize all the questions into words and convert them into lower case. Then compare words to a default dictionary to filter out the words that not exist. Next, apply spell correction and punctuation removing based on the data analysis I did above in order to clean the text.

For grammatical reasons, sentences will use different forms of a word, (e.g. do, did, done). Additionally, there are many derivationally related words with similar meanings, such as statistics, statistic and statistical. To reduce these kind of inflectional forms, I apply stemming and lemmatization to the text. I also restore abbreviations using regular expression, replace number-like string with real number. Below is one code example:

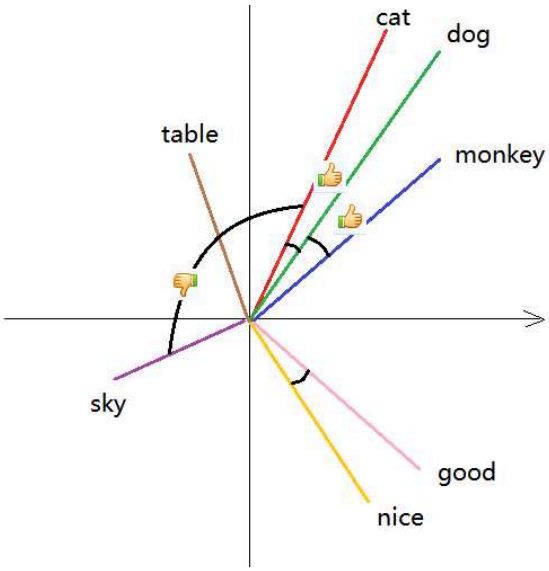
```
x=str(x).lower()
x=re.sub('[“”\('...\)\!/\^\"\.,:;,\-\?',' ',x]
x=x.replace('don't','do not').replace(',000,000','m').replace('$','dollar')
```

3.2. Feature

Some Kagglers describe that Kaggle competitions are 'feature engineering supplemented by model tuning and ensemble learning' [3]. Useful features can give better output. I include several features to achieve a better natural language processing system.

3.2.1. Word Embedding

Word embedding is a well known natural language processing technique to map words and phrases to vectors of real numbers. In other words, it is a mathematical embedding from one dimension per word to a continuous vector space. Figure 1 shows an example of word embedding. From this work, it is much easier for people to find synonym of one word using vectors.



Projection of the embedding vectors to 2-D

Figure1. word embedding example [4]

Word embedding has been proved to boost the performance in NLP problem such as syntactic parsing and sentiment analysis. During research, I find that GloVe (Global Vectors for Word Representation) performs excellent in this area [5]. It is an unsupervised deep learning algorithm for obtaining vector representations for words. There are some pre-trained GloVe word vectors available online. I use the one called glove.840B.300d.

I defined my own embedding features with this pre-trained vector. I decide to use Gensim, a python library to perform scalable statistical semantics work [6]. But this library only support word2vec format, I need to convert GloVe vectors in text format into the word2vec text format. The only difference between these two formats is that there is an extra header line in word2vec about the number of word and other information [7][8][9]. Therefore, I run the following command in terminal:

python -m gensim.scripts.glove2word2vec --input 'glove.840B.300d.txt'
--output 'glove_word2vec_convert.txt'

After loading the vector, I basically map all the words in training and test set into vectors of real numbers, then sum all of word embedding results and normalize it to unit vector to represent a sentence.

I compute various distance metrics, such as cosine, cityblock, jaccard, canberra, euclidean, minkowski, braycurtis distance for the results after applying word embedding. I also calculate skewness and kurtosis (Pearson) of the dataset. Finally, I save all these features into excel file for the future use.

3.2.2. Word sharing

According to [10], I implement the word share feature from the benchmark model. First I import a set of stop words from nltk.corpus. The stop words refer to the most common words in language such as 'and', 'also' and 'to'. This kind of words should be filtered out before

processing. After removing these common words, I compare each question pair and find the common words exist in both questions:

q1_words = set([word for word in q1_tokens if word not in STOP_WORDS])
q2_words = set([word for word in q2_tokens if word not in STOP_WORDS])
q1_stops = set([word for word in q1_tokens if word in STOP_WORDS])
q2_stops = set([word for word in q2_tokens if word in STOP_WORDS])

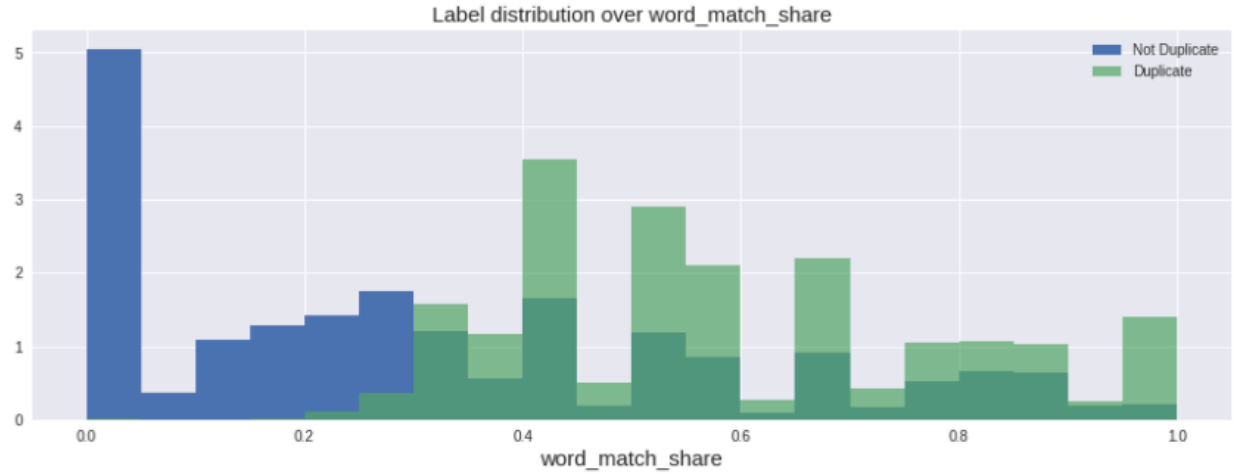


Figure2. [10]

Figure 2 shows that, in the training set, which the label is already given, it seems powerful enough to label question pairs as non-duplicate if the percentage of shared word is really low. But it is not good enough at identifying questions which are absolutely duplicate.

After that, I record the length difference information including common word intersection count, absolute differences of character length and word length, the ratio of longest common substrings in two sequences and so on.

3.2.3. Fuzzy string matching

There is a library called fuzzywuzzy [11][12], it is for string matching and depends only on the difflib python library. It uses Levenshtein distance to calculate the differences between two sequences.

I use *fuzz.ratio* to find string similarity.

There is attempts to account for partial string matches better. Function *fuzz.partial_ratio* uses the shortest string (length m) against all m-length substrings of the larger string and returns the highest score. In other words, we have a shorter string in length m and a longer string in length n, it is more interested in the score of the best matching length-m substring.

It is possible that two similar strings are out of order, so the result will be better if sort the strings first. There is more than one method in fuzzywuzzy library, and I use these functions to analyze data: *token_sort_ratio, token_set_ratio, partial_token_sort_ratio* and *partial_token_set_ratio*.

3.2.4. Magic feature

Quora [13] announce that, *'Therefore, we supplemented the dataset with negative examples. One source of negative examples were pairs of "related questions" which, although*

pertaining to similar topics, are not truly semantically equivalent.' From this, I can tell that the question pairs are made up of related questions, which means they may share a same tag/label. For example, both questions are related to environment.

There is a famous method called magic feature provided by Jared Turkewitz [14]. Turkewitz suggested that the duplicated questions tend to be the same group. That is, the higher frequency of one question occurrence, the more probable that the question pair is duplicate, no matter what question is paired with it.

This method gives me 0.00238 and 0.002 gain in private and public leaderboard respectively.

3.3. Model choosing

I implement two models in total, I will explain them below.

3.3.1. LSTM

Figure 3 shows my LSTM model. After separating training data into question1 and question2, map all the questions into number vectors using the function written by Ahmet Erdem [2]. Now question1 and question2 are passed into model as Input_1 and Input_2. Input_3 is the data processed by the features I mentioned above. In the whole model, I add 3 dropout layers, apply batch normalization three times and add Gaussian noise to the model.

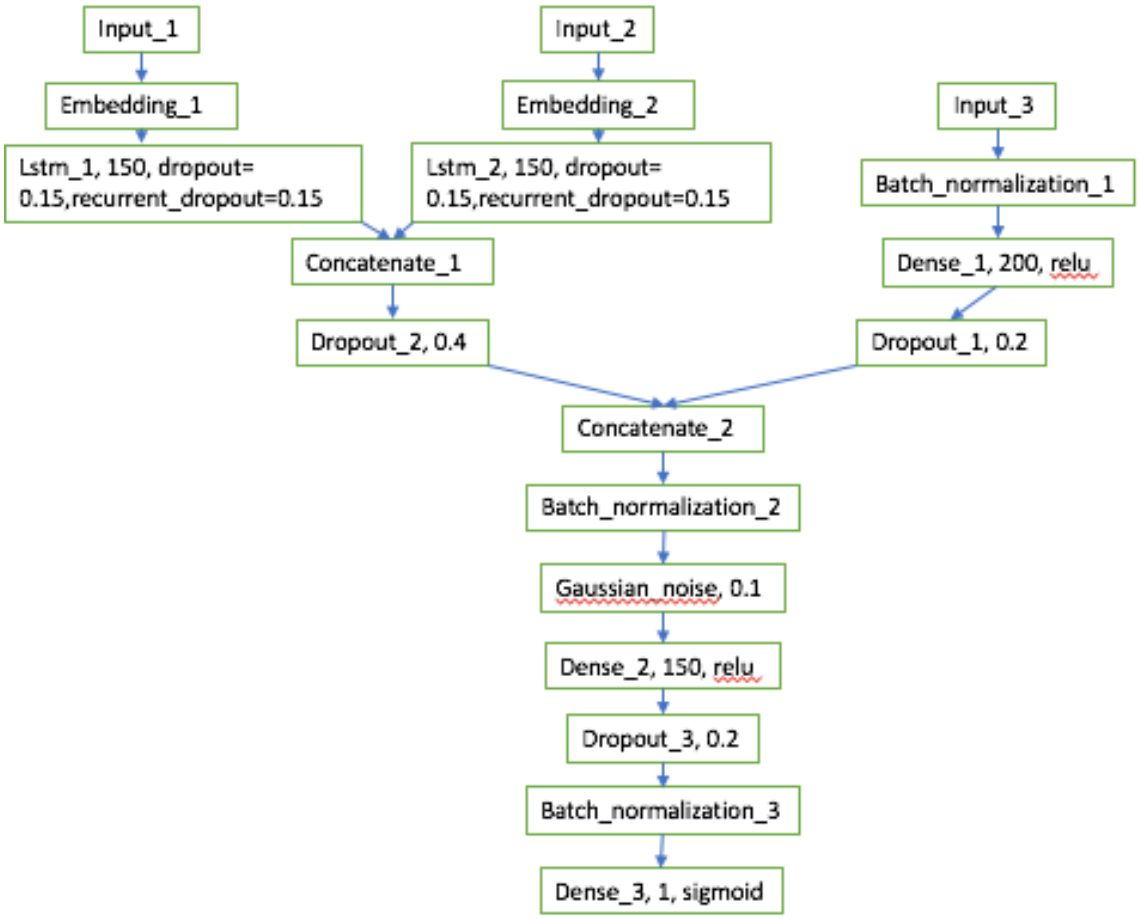


Figure 3. LSTM model schema

3.3.2. XGBoost

XGBoost is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable, it is very famous among Kagglers.

I just simply run XGBoost on data using its default function: *xgboost.train.*

3.3.3. Stacking with StratifiedKFold [3][15]

Ensemble learning is the technique of combining different models, and stacking is one method of that [3]. Applying stacking to training data can reduce both bias and variance of the final model and reduce the risk of overfitting. Figure 4 represents how stacking works. For example, I split training data into 5 folds. For each model, I run 5 iterations to traverse each fold and train 4 folds to predict the rest fold.

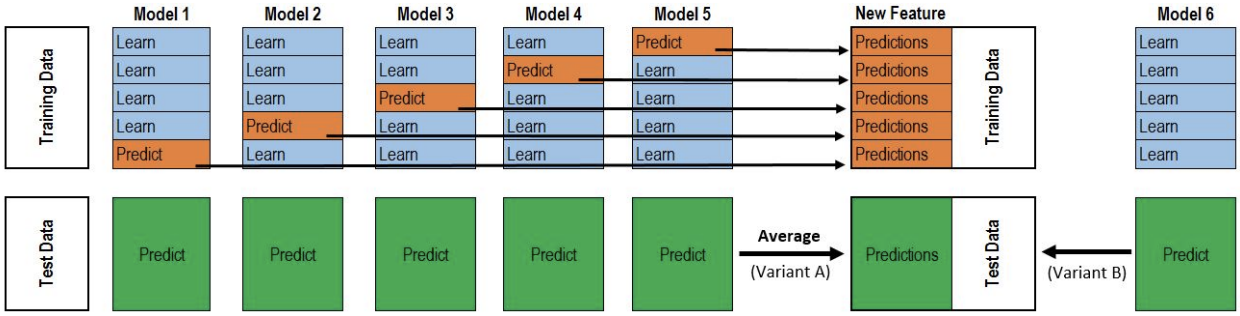


Figure 4. How stacking works [3]

In order to divide dataset reasonably, function StratifiedKFold is used. StratifiedKFold can split dataset into k stratified folds, each set contains approximately the same percentage of samples in the complete set. In other words, it split data while preserving the percentage of samples for each class.

In my case, I split training set into 5 folds, and apply 2 models (XGB and LSTM) to the data.

3.4. Post processing

One suggestion from Kaggle said that we need to rescale the training set and test set. After analyzing the training set, I can see that it has 37% positive class in training set, which means 37% question pairs in training set are labelled as duplicate. Without perform any rescaling, the log loss value is 0.55 for test set on public leaderboard. Using the following equation,

$$r = \frac{logloss + log\,(1-p)}{log\,(\frac{1-p}{p})}$$ [16]

p represents the positive class of training set; calculated r is positive class in test: 16.5%. This reveals that the distribution of training and test set is different. Therefore, I do need to apply rescaling to convert training predictions to test predictions.

In addition, to avoid oversampling during rescaling, I use the following method provided on Kaggle:

$$let\ a = \frac{0.165}{0.37},\ b = \frac{1-0.165}{1-0.37}$$

$$f(x) = \frac{a \times x}{a \times x + b \times (1 - x)}$$

4. Result

In order to complete this task, I have 8 files in total (Figure 5). I compile all the code on GPU and total compiling time is approximately 20 hours. (Some files can run in parallel).

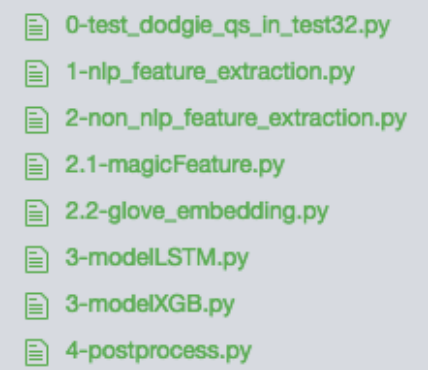


Figure 5.

There are 3307 teams on the leaderboard, the highest score is 0.11579. I decide to use Ahmet Erdem's code [2], I try to debug the code first and compile his code. Erdem is at 23rd on public leaderboard, the grade is 0.12988.

For my own implementation, after I add the features I mentioned in this report and run LSTM model, the highest score I can reaches 0.13064, the rank drops to 26th. For the single XGBoost model I defined, the score is only 0.15455, the rank is around 297th. But if I combined LSTM and XGB with stacking method, the log loss value reaches 0.12890, 20th position on public leaderboard. This is my highest score, even exceed Erdem's. What's more, the magic feature gives me around 0.002 gain on leaderboard.

However, when I add self-defined GloVe feature to the program, the result drops down to 0.2119. It shows there may be some overfitting issues.

Log Loss	Private Score in LB	Public Score in LB
LSTM	0.13295	0.13064
XGBoost	0.15749	0.15455
LSTM+XGBoost	0.13149	**0.12890**
LSTM+XGBoost+self_defined Glove	0.21560	0.21190

5. Conclusion and Summary

Features are very important in this kind of competition, we should try as many features as we can since different feature can cover different aspect. But some features may introduce overfitting problem. There also is tradeoff between training speed and performance. Therefore, it is vital to pick up the most significant features.

As for models, due to the time limitation, I only try two models. And my stacking method only combines the two models equally together. I could apply some method to combine different models with calculated weights.

6. Reference

[1] https://www.kaggle.com/c/quora-question-pairs/data
[2] Erdem, A. (2017, May). 24th Place Solution Repo. Retrieved November 11, 2017, from https://www.kaggle.com/c/quora-question-pairs/discussion/34534
[3] W. (2016, May 10). How to rank 10% in your first Kaggle competition. Retrieved from November 11, 2017, from https://dnc1994.com/2016/05/rank-10-percent-in-first-kaggle-competition-en/
[4] Li, E. (2015, August 26). Word Embedding. Retrieved from November 14, 2017, from https://www.zhihu.com/question/32275069
[5] Pennington, J. Socher, R. Manning, C. D. (2014, August). GloVe: Global vectors for word representation. Retrieved from December 1, 2017, from https://nlp.stanford.edu/projects/glove/
[6] Unknown. (n.d.). Retrieved from December 1, 2017, from https://radimrehurek.com/gensim/tutorial.html
[7] Unknown. (n.d.). scrpts.glove2word2vec-convert glove format to word2vec. Retrieved from December 1, 2017, from https://radimrehurek.com/gensim/scripts/glove2word2vec.html
[8] Unknown. (n.d.). Retrieved from December 1, 2017, from https://github.com/3Top/word2vec-api/issues/6
[9] Oakes, J. (2016, May 3). Retrieved from December 1, 2017, from https://github.com/jroakes/glove-to-word2vec
[10] A. (2017, March). Data Analysis & XGBoost Starter. Retrieved from November 10, 2017, from https://www.kaggle.com/anokas/data-analysis-xgboost-starter-0-35460-lb
[11] Unknown. (2011, July 8). FuzzyWuzzy: Fuzzy String Matching in Python. Retrieved from November 26, 2017, from http://chairnerd.seatgeek.com/fuzzywuzzy-fuzzy-string-matching-in-python/
[12] L. (2017, August 4). When to use which fuzz function to compare 2 strings. Retrieved from November 26, 2017, from https://stackoverflow.com/questions/31806695/when-to-use-which-fuzz-function-to-compare-2-strings
[13] Csernai, K. Iyer, S. Dandekar, N. (2017, January 24). Fisrt Quora Dataset Release: Question Pairs. Retrieved from November 11, 2017, from https://data.quora.com/First-Quora-Dataset-Release-Question-Pairs
[14] Turkewitz, J. (2017, April). Magic Features. Retrieved from November 15, 2017, from https://www.kaggle.com/jturkewitz/magic-features-0-03-gain
[15] Unknown. (n.d.). Cross-validation: evaluating estimator performance. Retrieved from December 1, 2017, from http://scikit-learn.org/stable/modules/cross_validation.html
[16] Thaler, D. (2017, April). How many 1's are in the Public LB? Retrieved from November 18, 2017, from https://www.kaggle.com/davidthaler/how-many-1-s-are-in-the-public-lb