Machine Learning Engineer Nanodegree
Capstone Project

Shun-Wen Chang

May 11th, 2017

I. Definition

Project Overview

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.
Over 100 million people visit Quora every month, so it's no surprise that many people ask similarly worded questions. 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.

Problem Statement

The goal of this project is to predict which of the provided pairs of questions contain two questions with the same meaning. I will be tackling this as a natural language processing problem and apply supervised learning techniques to classify whether question pairs are duplicates or not. Doing so will make it easier to find high quality answers to questions resulting in an improved experience for Quora writers, seekers, and readers.

Metrics

Prediction results are evaluated on the log loss between the predicted values and the ground truth. 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.

Since this is a Kaggle competition project, I will take the leaderboard score as my final evaluation. The leaderboard score is actually log-loss of the test data, which are unseen to the public. However, for my training process, I will also use the model accuracy as the screening metric to select the best training model, and then use log-loss to fine tune the parameters. The log-loss is a great metric for this project because we are predicting the probability or label being either 0 or 1. Logistic loss, aka cross-entropy loss, serves exactly this purpose.

II. Analysis

Data Exploration

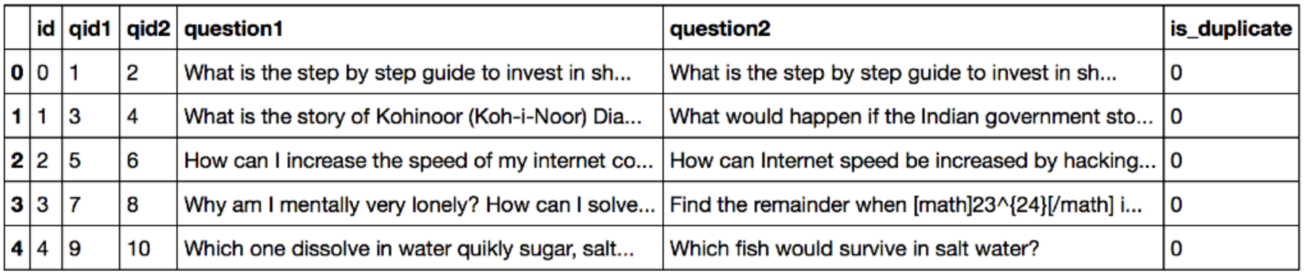
The training data and test data from this problem are provided on Kaggle website: https://www.kaggle.com/c/quora-question-pairs/data

Training data set is 21 MB and test data set is 112 MB. There are much more entries in test data set.

There are 6 data fields in the training data:

- id - the id of a training set question pair
- qid1, qid2 - unique ids of each question (only available in training data)
- question1, question2 - the full text of each question
- is_duplicate - the target variable, set to 1 if question1 and question2 have essentially the same meaning, 0 otherwise.

Let’s look at the first 5 lines of training data:

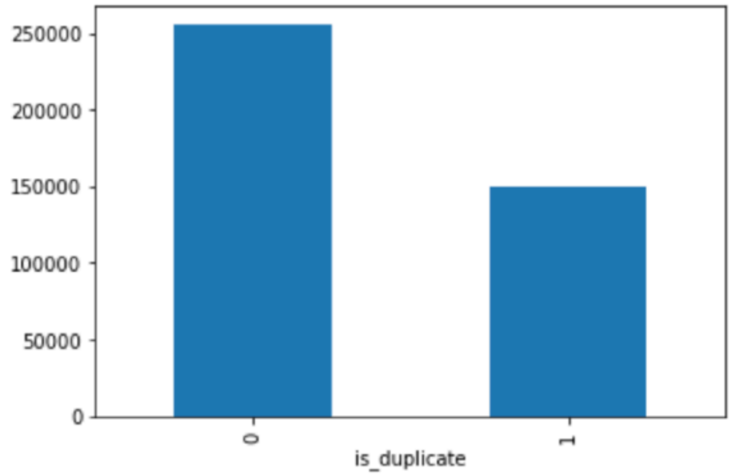


	id	qid1	qid2	question1	question2	is_duplicate
0	0	1	2	What is the step by step guide to invest in sh...	What is the step by step guide to invest in sh...	0
1	1	3	4	What is the story of Kohinoor (Koh-i-Noor) Dia...	What would happen if the Indian government sto...	0
2	2	5	6	How can I increase the speed 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

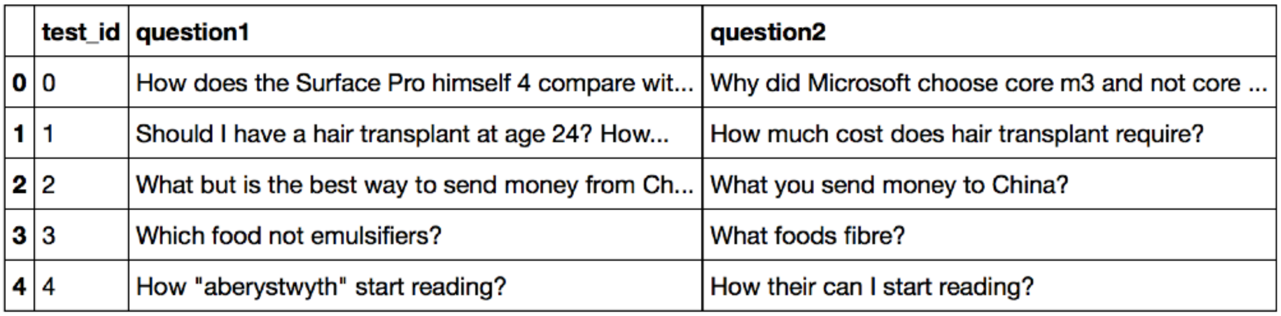
Everything looks fine at this point. I will deal with NaN values later.

Some statistics of this data set:

- Number of training data: 404290



- Ratio of duplicate question pairs: 36.92%
- Total number of questions: 537933



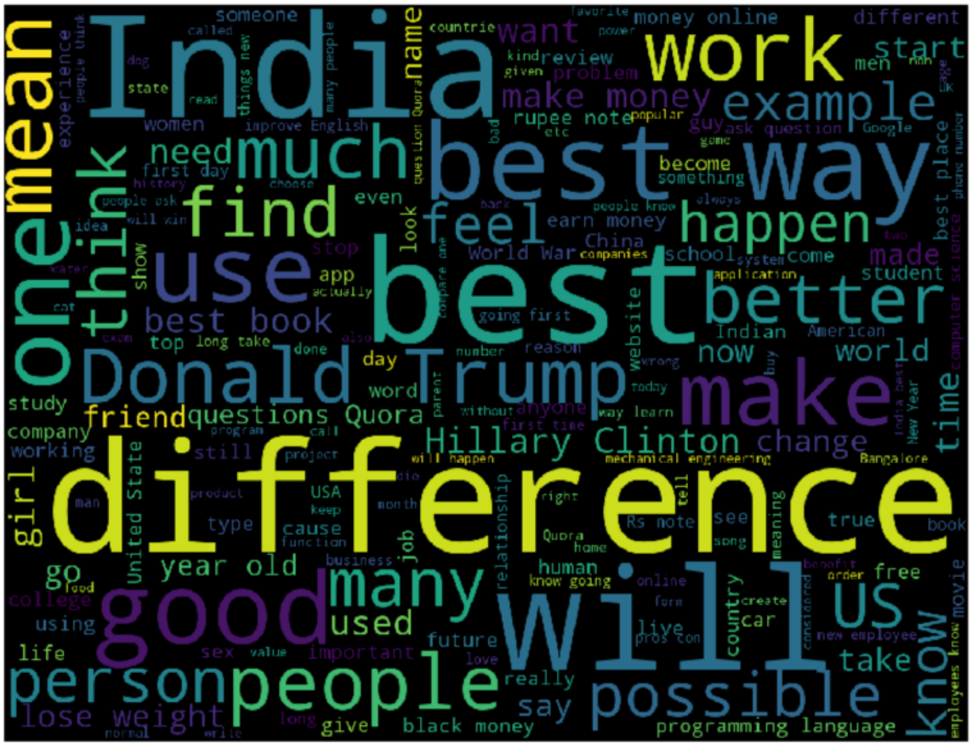
	test_id	question1	question2
0	0	How does the Surface Pro himself 4 compare wit...	Why did Microsoft choose core m3 and not core ...
1	1	Should I have a hair transplant at age 24? How...	How much cost does hair transplant require?
2	2	What but is the best way to send money from Ch...	What you send money to China?
3	3	Which food not emulsifiers?	What foods fibre?
4	4	How "aberystwyth" start reading?	How their can I start reading?

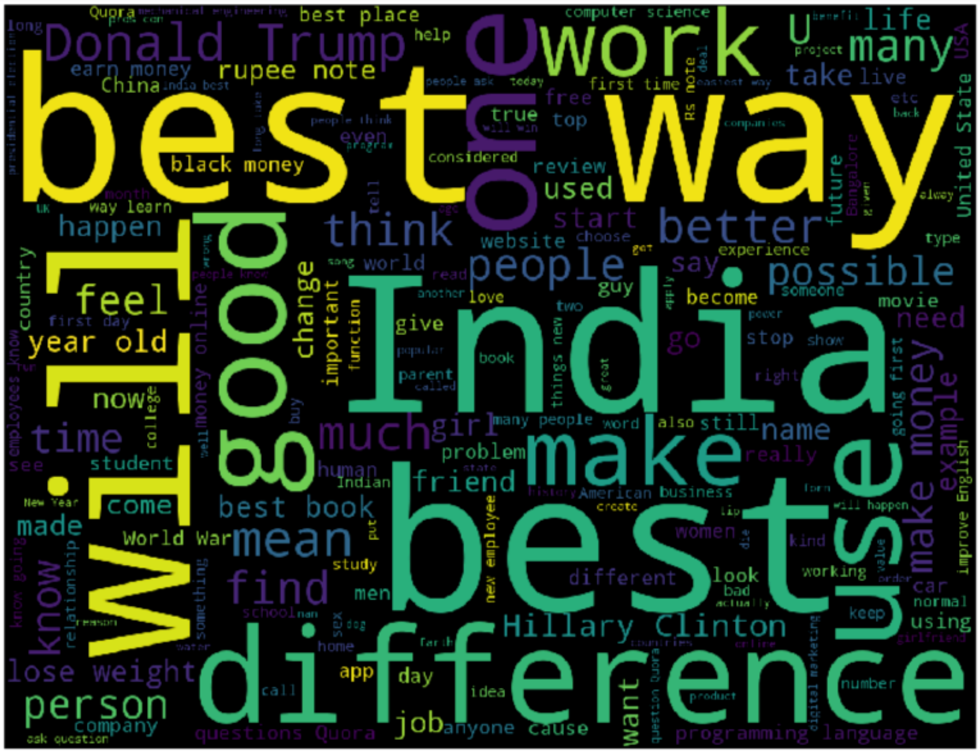
- Number of questions appearing multiple times: 111780

Now let’s take a look at the first 5 rows of test data set:
Again, everything looks fine so far. There is no qid1 and qid2 in test data, only test_id for labeling the question pairs. I will deal with NaN values later in this project.

Another interesting way to get a sense of the data is to generate a word cloud graph, which shows a bunch or words in a graph. The bigger the word font the more frequent it shows up in the word corpus. Here I am separating question1 and question2 into two corpuses and using *wordcloud* library to generate plots.

question1:


India
best
way
difference
will
work
mean
one
think
use
much
find
feel
happen
example
better
make
Donald Trump
good
people
person
possible
many
used
know
US
Hillary Clinton
change
questions Quora
friend
lose weight
make money
best book
need
start
want
name
someone
money online
different
year old
girl
United State
black money
programming language
take
future
world
time
now
rupee note
earn money
World War
China
school
become
something
student
Indian
American
website
day
word
study
company
working
still
type
cause
go
college
using
life
sex
important
really
say
give
free
car
country
live
movie
book
true
see
human
job
relationship
mechanical engineering
Bangalore
best place
computer science
New Year
first day
women
improve English
men
ask question
Google
app
look
stop
reason
top
even
come
made

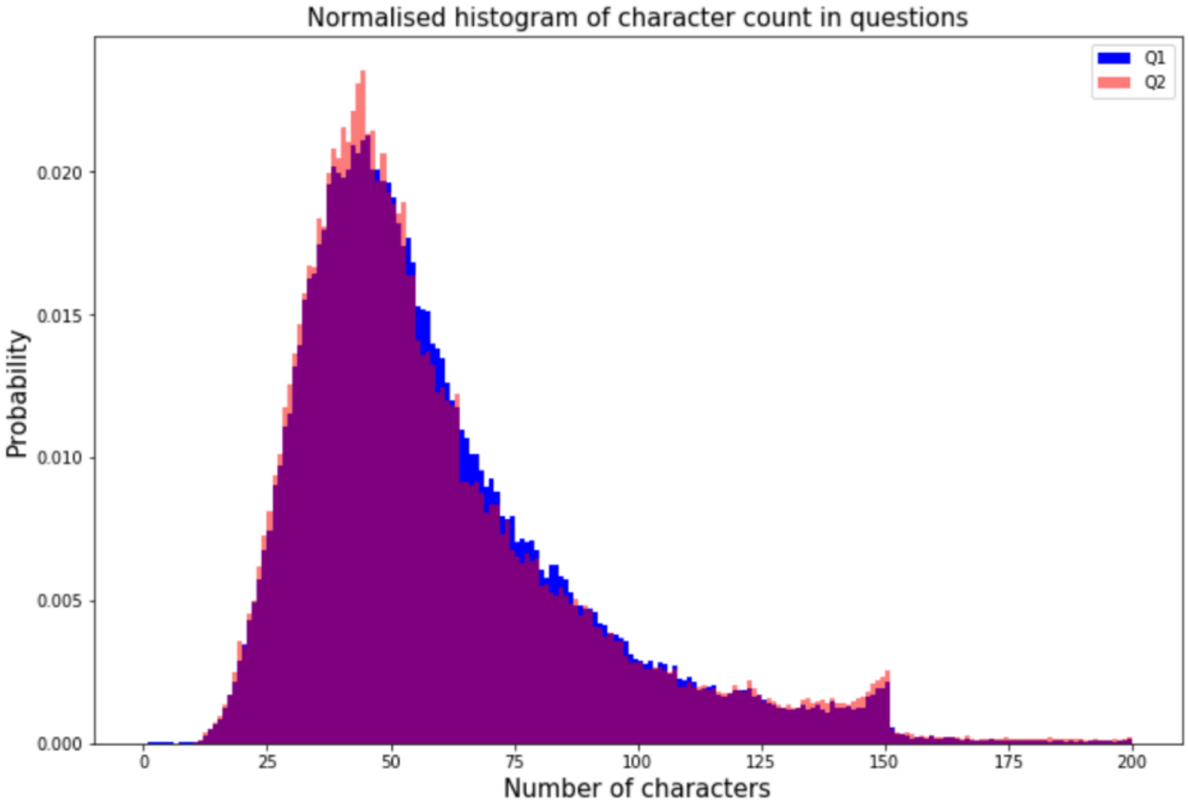

best way
India
best
difference
will
good
one
work
use
Donald Trump
think
better
people
possible
make
much
mean
find
feel
happen
year old
time
know
person
lose weight
Hillary Clinton
friend
best book
made
come
student
World War
many
life
take
United State
rupee note
earn money
China
black money
best place
computer science
start
used
review
considered
true
top
free
website
world
change
important
girl
problem
make money
example
improve English
need
movie
become
say
guy
stop
name
still
really
different
look
kind
keep
using
car
normal
job
anyone
cause
questions Quora
programming language
want
app
day
idea
company
ask question
study
men
home
business
American
Indian
human
now
country
go
give
parent
book
function
experience
type
someone
USA
many people
word
things new
going first
future
Bangalore
live
help
Quora
mechanical engineering

question2:

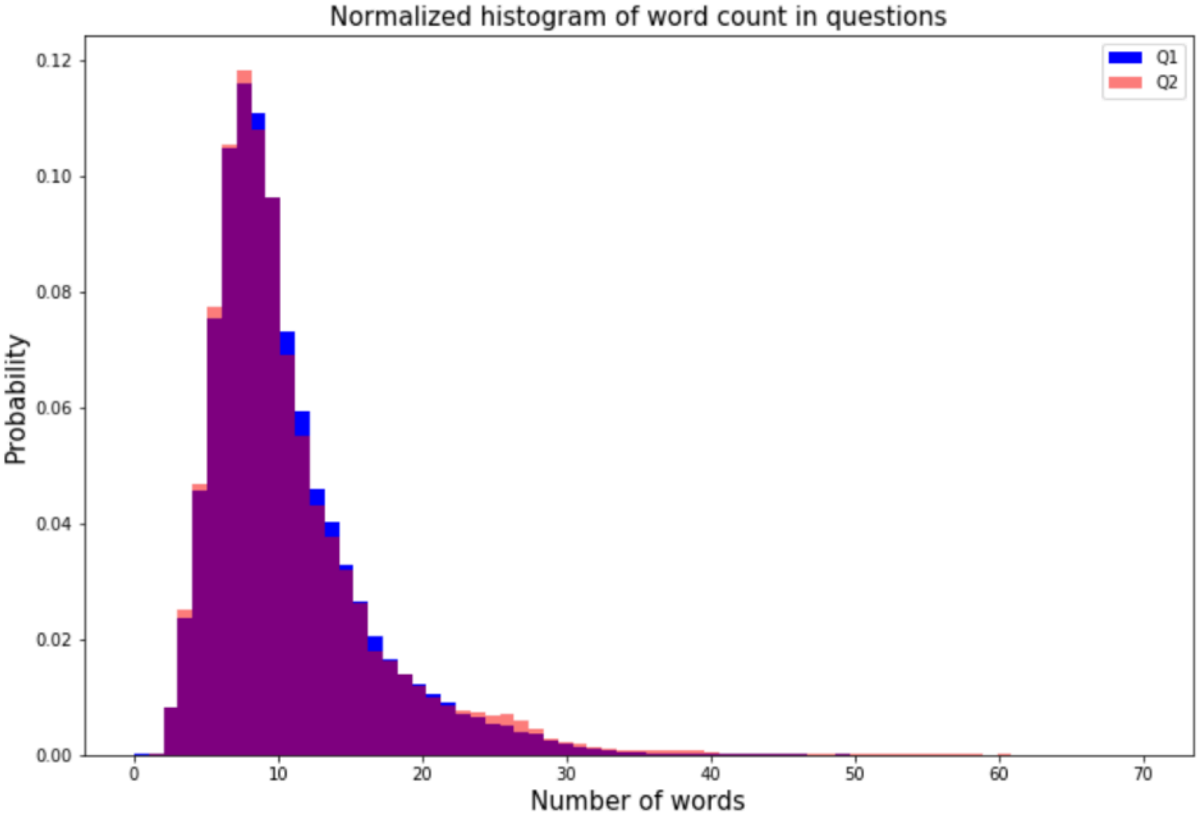
Take a close look at these two pictures, we notice that there are several words appearing in both graphs. For example: best, difference, way, good, make…etc. Most of them are common words, meaning that they show up almost everywhere in our daily conversation. However, we can see some of the words are special, like “Donald Trump”, “India”. This tells us what the most frequent questions are possibly about.

Exploratory Visualization

The first plot shows a normalized histogram of character count in both question1 and question2. Blue color represents question1 and red color represents question2. From the graph we can see that the character counts distributions are almost the same for question1 and question2. They all peak at around 45 characters and have an average of 55 characters. One interesting feature is at 150 characters there is a slight increase in numbers and then an abrupt drop. This may be from the rule of Quora website limiting the amount of words user can type in, with the exception of math equations or hyperlinks, hence explaining the little tail above 150 character count.

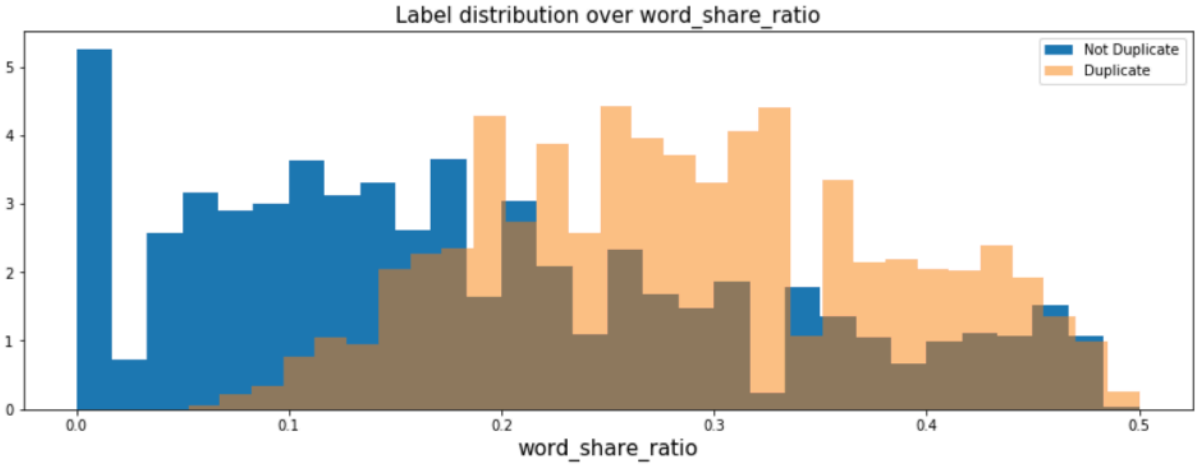


Second graph shows the normalized histogram of number of words from question1 and question2.

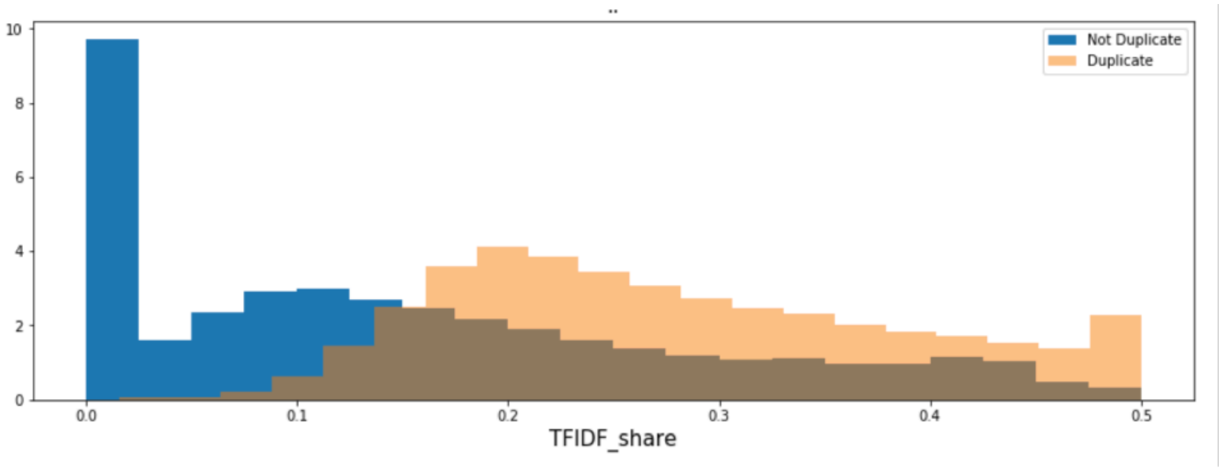


Again, there is almost the same distribution in both question1 and question2. The number of words in questions peak at 10 words. Most people ask questions with roughly 10 words, which is expected.

The third graph shows the bar plot of word_share_ratio. Word_share_ratio is an extracted feature from the training data, which will be discussed later in the next section. The larger the ratio the more overlapped the questions pair is. In this graph, I use beige color for duplicate question pairs and blue color for non-duplicate question pairs. As we can see, duplicate question pairs generally have more words shared between them. This observation is what we would expect.



The final graph I generated is the TFIDF word share ratio. TFIDF word share ratio is also a feature extracted from the training data and will be discussed in the next section. For now, you can think of it as similar to word_share_ratio, the bigger the number, the more similar the two questions are.



Based on these graphs, I am confident that I can use these as features for my classification algorithm.

Algorithms and Techniques

Since this is a binary classification problem, I will be using supervised learning algorithms. First I will clean up the strings by removing leading and trailing white space and converting to lower case. Then I will extract features from the attribute I am given on the data. These features include: 1. word count 2. character count 3. word share 4. TF-IDF share. The algorithms I will be implementing are: 1. Random forest 2. Logistic regression 3. Decision trees 4. K nearest neighbors 5. Naive bayes 6. Support vector machine 7. Gradient boosting. I will use validation data set to compare these 7 algorithms and find the best one, then do fine tuning on the best algorithm to get optimal accuracy.

Logistic regression is a regression model where the outcome is binary, and the model predicts the probability (0 to 1) of the outcome.

Decision trees is a model which uses a tree to do binary classification at each node (feature), and at the leave of the tree a label prediction is returned.

Random forest is by using an ensemble of decision trees to reduce overfitting of one tree model. The training algorithm for random forest applies the general technique of bagging, which means selecting a random sample of training data and fitting trees to these samples. Finally we average the result of all trees to make prediction.

K-nearest neighbor model is using the average of k nearest data points to predict the testing data value. The “distance” her is usually 2D Euclidean distance.

Naive Bayes classifier is a family of simple probabilistic classifiers based on applying Bayes’ theorem with naive independence assumptions between the features.

Support vector machine is an algorithm which outputs an optimal hyperplane to separate labeled training data.

Finally, gradient boosting model is similar to random forests, with the difference being how we train them. For gradient boosting, we assign prediction score in each leaf instead of a binary value, and during training we add one new tree to the ensemble at a time and do the optimizing.

Benchmark

The benchmark model in my problem is the random forest model. The reason is that random forest is easy to implement and naive enough to be compared with other algorithms. There is no published result for this problem, but since it is a Kaggle competition, I can get a sense of how my model is performing by looking at the leaderboard score and ranking.

III. Methodology

Data Preprocessing

First I strip the string from leading and trailing white space, and then I convert it to all lower case. I also remove NaN values in character_count by replacing them with zeros.

There are a total of 6 features. Question1 number of words, Question1 character count, Question2 number of words, Question2 character count, word share ratio, TFIDF vector share ratio. The first four are extracted using basic python string functions.

Word_share is calculated as the number of words appearing in both question1 and question2 divided by the total number of words in question1 and question2. The result is a normalized value measuring how much word-overlap is between question1 and question2.

For TFIDF factors, I did not use the sklearn TFIDF library. Instead, I wrote a similar function implementing this TFIDF idea so that I don’t feel I am using a blackbox function, and it gives me more freedom to do the TFIDF vector share calculation. First I combine all questions into one big corpus, and then calculate weight for each word. The weight is calculated by the inverse of word count plus epsilon (=5000) such that the less common words get higher weights. I ignored words that only appear once in the whole corpus for the reason that it may be typos. Then I calculate the tfidf_word_share_norm using share_weight divided by total_weight. Share_weight is the total weight of the words appearing in both question1 and question2. Total_weight is the total weight of every words in either question1 or question2. The result is a normalized TFIDF_weight_share. Since TFIDF discounts on the common words that appear universally in our language, this measure represents more closely to how similar two questions are.

Implementation

I have tried 6 supervised learning models excluding the benchmark model. Except the Naive Bayes model, I normalized all feature values before doing model fit. The following are implementation of each model:

1. Logistic regression: I used GridSearch cross validation to find best parameters (‘C’, ‘penalty’), which are (‘1000’, ‘l2’).
2. Decision tree: For this one I just simple fit and predict functions of sklearn library.
3. Support Vector Machine: I did GridSearch cross validation to find best parameters (‘C’, ‘kernel’), which are (‘0.001’, ‘sigmoid’). I set max iteration to 500 to avoid “out of memory” issue.

4. K-Nearest Neighbors: I also did GridSearch cross validation to find best parameters (’n_neighbors’, ‘weights’), which are (‘8’, ‘uniform’).
5. Naive Bayes: Naive Bayes features work better if their values are integers. Since Naive Bayes is based on probability, there is no need for feature value normalization.
6. Gradient Boosting (XGBoost): My setting for parameters are {‘objective’: ‘binary:logistic’, ‘eval_metric’: ‘logloss’, ‘eta’: 0.05, ‘max_depth’: 3} for the beginning. I did 200 rounds of boosting and if there’s no improvement after 50 rounds the training will stop.

One challenge I had during my implementation is, after I did all the model training and got 0.39 logloss for my best result, I was still thinking if somehow I can improve the model by adding more features. The first thing I could think of is adding TFIDF vectors as my new features. I started by using sklearn.feature_extraction.tesxt.TfidfVectorizer to extract TFIDF vectors from my question strings. I ended up with 10,000 new features for each question. Since training models with such huge number of features for this simple problem doesn’t make much sense, I use SVD to reduce these 10,000 features down to 20. Then I tried two different ways to incorporate these features. First one is calculating the “distance” between question1 and question2 with these vectors; second one is just leave them as they are. Neither of these two approaches gave me any better accuracy in terms of log-loss score. I was frustrated but still continued on trying different scales of reduction. I tried 20, 50, and 100 TFIDF features but none of them give me any impressive result.

Refinement

Based on the Kaggle leaderboard score for the 6 models, XGboost gives the best result. Therefore, I started playing around with ‘eta’ (learning rate) and ‘max_depth’ (model complexity). The numbers I have tried are :

eta	0.01	0.02	0.1	0.5
max_length	3	4	5	6

I found that ‘eta’=0.02 and ‘max_depth’=4 gives me best result without overfitting. When my validation data has higher logloss than my training data, I know there is overfitting. After finding the optimal parameters, I did 500 rounds of boosting to get my final logloss number, which is 0.39. At one point, this number gives me 54% ranking on Kaggle leaderboard.

IV. Results

Model Evaluation and Validation

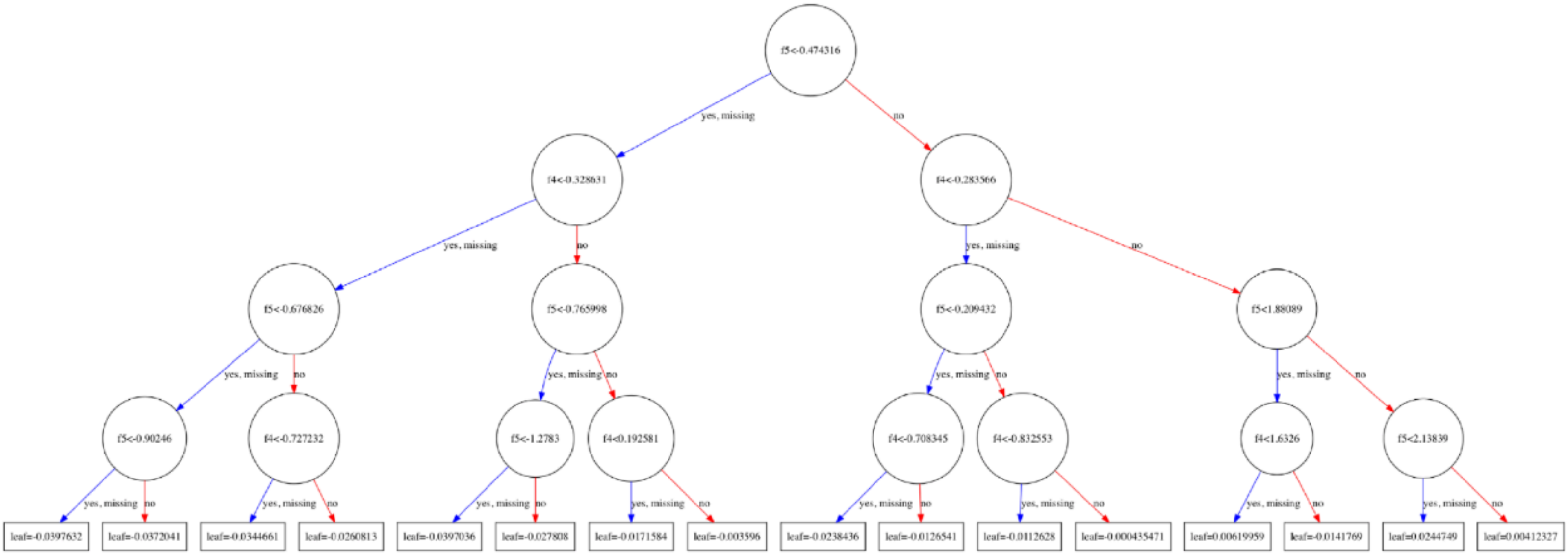
The final model I used is XGBoost (Gradient Boosting or Boosted Tree), which is similar to random forest, but with a better performance in my problem. I have used validation data set to do make sure there is no overfitting and to find the best parameters within reasonable training time. The model is pretty robust because a boosted tree uses many trees to do the training, and unlike random forest, it uses scores in the leaves instead of binary classification. Also, the Kaggle competition score is based on unseen data that are not accessible to anyone. If I can get similar logloss value in my training and the “unseen” testing data in the competition, I am confident that my model is valid.

Justification

Yes the final result is significantly better than the benchmark model. Although the idea behind random forest and boosted trees is the same, the training process is different. I think boosted trees is a more robust model and hence should give better result. In my project, benchmark model has log-loss 0.55 while XGboost model has log-loss 0.39. I believe the final result is good enough for solving this problem. It is not 100% accurate, but should be above 80%. For a problem like this, identifying more than 80% of the duplicate questions is pretty good enough for saving users time.

V. Conclusion

Free-Form Visualization



Using plot_tree() function we are able to visualize the boosted trees in the XGBoost training model. One interesting phenomenon I observed from this tree is that only two features are dominating the final decisions (f4 and f5). I would guess these two features to be word_share and TFIDF_word_share. Intuitively they are the most important features in our classification.

Reflection

At first this problem seems to be a simple linear regression classification problem. To me the tricky part is to select what I want to use for my features. I first thought about the number of words and character length, and then the amount of the same words shared between two questions. Since this is obviously a NLP related problem, I thought of using TFIDF to process the sentences. Finally I ended up with 6 features. I tried 7 different supervised machine learning algorithms. Gradient boosting gave the best result. After that I fine tuned the parameter and got 0.39 log-loss (ranked top 54% on Kaggle leaderboard). As mentioned before, I have tried to add more features extracted from TFIDF vectors, but did not get any significant accuracy gain. My reflection on this is: adding features is not always better. Because in my original features I already implemented my own TFIDF weighted word share, adding more TFIDF vectors later may not improve my model accuracy. It may still improve a little, as I have observed, but not by much. I would probably not want to spent an extra week of training time for a 1% accuracy improvement.

Improvement

To improve this project in the future, I will do more sentiment analysis. Right now I only have done simple string preprocessing and then strip out the words. In the future, I will spend more effort on natural language processing techniques. I may try extracting the punctuation, upper/lower letter, special characters, foreign languages, math symbols, emojis…etc, for more detailed analysis.

If computing power allowed, I can also try doing deep learning neural network on this problem. Since we can extract over a million TFIDF vectors, it may be interesting to do convolutional neural networks with them and see what comes out.