DS-GA-1003

Spring 2018 Machine Learning

# Quora Question Pairs: Identify Question Pairs That Have The Same Intent

Yu Xiong(yx1201), Jin Han(jh5695), Sijia Liu(sl6496)

May, 2018

**Abstract**

*As large text data are generated and recorded in today's world, our ability to detect semantic similarity and equivalence in sentences become more and more important. This technique can be applied in many fields, including social media and research. Detecting similarity can be done in many ways. The simplest is to match the word occurred in the two sentences. However, this approach may not be able to capture the dynamic and subtlety nature of language. In this paper, we took the Quora question pair dataset and developed an empirical model using machine learning methods to estimate whether two questions are equivalent to not. We used Gloves and TF-IDF for word embedding and examined the similarity of questions by KNN, Random Forest and XGBoost. The objective of modeling was to minimize the log-loss of predictions on duplicacy in the testing dataset. Finally, we achieved significant lift against the baseline mode.*

## 1. Introduction

In this digital world, the number of new documents and information are increasing exponentially. New research findings and reports are released. Hundreds of news articles, stories and tweets are generated every day. Thousands of facebook posts and events every day. Numerous new questions being posted on platforms like Quora and StackOverflow. The information explosion does have its own good. New ideas are more likely to come across one's mind when one is immersed in an environment with diverse information. However, for people who are looking for an answer for a specific question, the huge amount of information can be a barrier. For example, when people are searching on StackOverflow looking for help with their code, they might find multiple opened questions related to their bug. It would be time consuming to go over all posted questions to find the best solution. People might also distracted unrelated information when the they need to go over all related question to search for an answer.

This is a bad user experience for both writers and seekers, as the answers get fragmented across different versions of the same question. For the real-world problem out there, we hope AI could solve it.

### 1.1 Business Problem and Motivation

As the amount of available data grows in the digital age, the problem of managing the information becomes more difficult. The information explosion will likely lead to information overload. People might not be able to efficiently allocate their time in making the right decision on an issue or finding the answer to their question when they have too much information about the issue. To solve this problem, we can have a similarity detection system to test across all posted contents and detect the ones that are duplicates in their meaning.

This mechanism can be used in many scenarios. For example, models for detecting similarity sentences could be used to better organize and consolidate the questions or answers for platform such as Yahoo or Quora. It would be easier for the users to find high quality answers to questions.Such model could also be applied to improve the performance of smart speakers such as siri and Amazon Echo Dot. Moreover, A similarity detection system would be crucial for detecting the plagiarism or duplicated ideas in the research field. In those scenarios, a mature product should be an information system that automatically filters, merges or flag the publication that are very similar in the content.

In this paper, we are trying to build a model that could recognize not only common pattern in two questions, but when two questions use different words and phrases with the same semantic meaning. Our model tries to learn these patterns to achieve better prediction.

### 1.2 Previous Works

As sentence similarity detection widely used in some applications such as plagiarism detection or Question-Answer platform, there is extensive literature on measuring texts similarity. There are some other techniques that are related to our work. Some early approach for detecting sentence similarity uses word overlaps or surface-matching method. As Devrim Akca introduced the idea of word overlaps and surface-machining in his paper, this method is based on the number of words occurrence in both text segments(1). This technique relies on the assumption that if two texts are similar then they shares more duplicate words. In 2006,  Landauer proposed a corpus-based methods(2). The term co-occurrences in a corpus are operated by a singular value decomposition (SVD) for dimension deduction and generate a matrix T representing the corpus. Then, Vector similarity is then used to identify documents that are most related to the query. For now, more and more nature language process techniques are used on this problem to include semantic meaning in the sentences such as gloves, word2vec.

In this paper, we are going to combine semantic embedding, vector similarity and machine learning modeling together to predict the Quora question similarity.

### 1.3 Problem Formulation

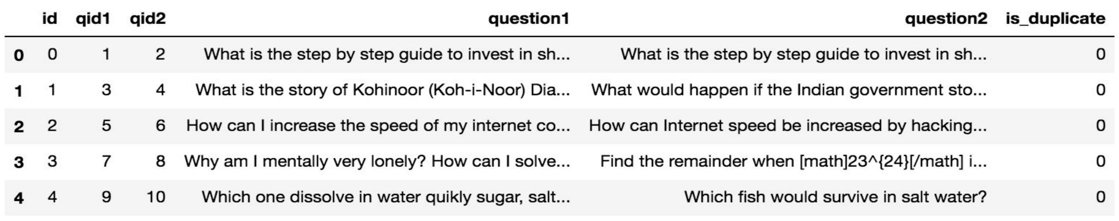
In this paper, we would like classify whether question pairs are duplicates or not by applying advanced machine learning techniques. More specifically, our problem is a binary classification problem in which we take two natural language questions and return a binary result duplicate/different.

More specifically, given a training dataset, we would like to derive a model that could minimize the log-loss on the test dataset. Details on our training dataset, which contains the features X, input information related to the two questions of interest and our label are explained in the section below.

## 2. Data Analytics

Kaggle provided the Quora Question Pair dataset collected by researchers from Quora, Inc, which contains 404,290 labeled samples and 2,345,796 unlabeled test samples for competition purpose. We are going to use the labeled samples for our classification problem.

### 2.1 Descriptive Statistics



| | 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 |

Figure 1: Quora Data Sets

Our dataset contains 404290 rows and 5 columns. As figure 1 shows, the first one column ***'ID'*** is a simple row index, then the next two columns ***'qid1,2'*** is the question ID number. Next two columns ***'question1,2"*** contain the natural language question pair in English respectively. The last column ***'is_duplicate'*** is our target variable, identifies if this question pair conveys the same information. It is a binary column which only contains 0, indicating that the question pair is semantically equivalent or 1, the question pairs ask about different things. There are 36.9% positive instance and 63.1% negative instance.

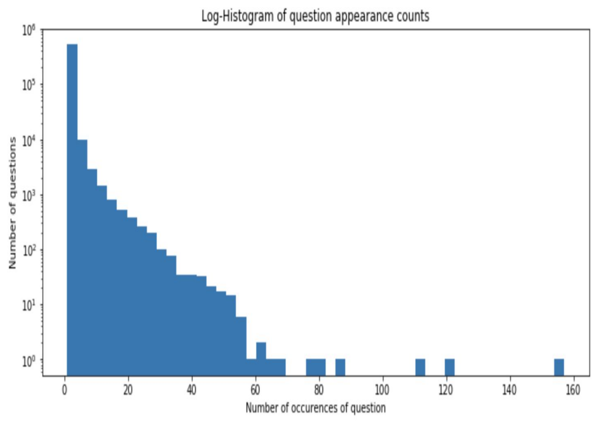


Figure 2: Occurrence Graph

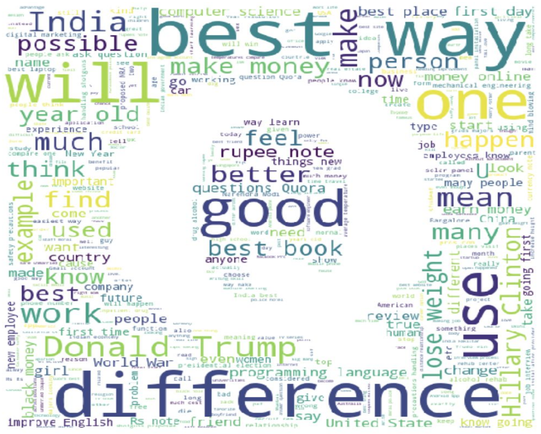


Figure 3: Word Cloud

Among the 537933 questions, the occurrence graph above shows that most of the questions only appear a few times, while some questions have large occurrences for more than 100 times. One of them even appears around 160 times. After scrutinizing the outlier, we found it out as follows:

The most frequent question is “What are the best way to lose weight?” And there are lots of similar questions were asked and paired with the most frequent one. It is interesting since we believe that there are many many people trying to find the answer about losing weight. The question ‘What are the best way to lose weight?’ can be seen as a standard content of this kind of problem and any other similar problem can be compared with this standard one to figure out weather they are of same intent.

We use a Word Cloud to see the most common words among all questions. Not surprisingly, the words such as “best”, “difference”, “way”, “use” seems to happen more frequently.

## 2.2 Data Cleaning

We find that the data is very clean and of good quality. There are only two null values in the second question. We delete the rows of null values directly since it doesn’t make sense to do numerical imputation and replace null questions with any other values. There are also a small portion of values are non-ASCII characters when tokenizing. Here we use ***.decode()*** function to convert all strings to ASCII for computer proprecessing.

It is worth to mention that, because of the subtlety of languages, different people might have different interpretation of the questions. The label 0/1 in the dataset provided might not be absolutely true. This could be a potential cause for the model error.

## 2.3 Feature Engineering

To distinguish whether two sentences are semantically equivalent, we start with simple properties of a sentence such as length and duplicated words. We will then dig in to the semantic properties of a sentence, ie. features that represent the actual meaning of a sentence.

### 2.3.1 Baseline Feature Engineering: Sentence Structure

Since our data are comprised of natural language sentences, we tokenized the words in each question. The simplest way of identifying two equivalents is to see how many words occurred in both questions. Intuitively, if both questions have mostly identical words, they are more likely to be identical.

From the standpoint of sentence structure, we think that sentence length might be relevant in identifying content. For example, a long sentence might be expressing a complicated logic while short sentences might be a straight-forward logic. If both sentences are around the same length, there is a higher chance that they are of similar meaning.

Moreover, we convert each questions to a vector of token counts using CountVectorizer in Sklearn. This function implements both tokenization and occurrence counting of the sentences/questions. And it produces a sparse representation of the counts. We calculated Cosine similarity using the tokenized vector to investigate the similarity of two questions.

Features added in this step including:

- Length of question 1: ***q1_len***
- Length of question 2: ***q2_len***
- Absolute difference in the two sentences' lengths: ***abs_len***
- Number of duplicate words in two questions: ***duplicates***
- Percentage of duplicated words in two questions:***dup_percent***
- Cosine similarity calculated using the tokenized question: ***cosine_sim***

### 2.3.2 Advanced Feature Engineering: Sentence Meaning

In terms of the meaning of the sentence, we use word embedding. There are two major methods for learning vector space representation of words: local context window methods, which does better in word analogy task since they are trained on a local window but does not utilize the statistics well; matrix factorization methods, which efficiently employs the statistical information but did poor in word analogy task.

Since our main goal is to detect sentence meaning similarity, our embedding method is generated using pre-trained Glove embedding(3), which performs much better in word analogy.

There are two steps:

1. **Tokenizing the questions contents.** Here we used NLTK(Natural Language ToolKit) package to complete it, which is the most commonly used suite of packages for symbolic and statistical natural language processing for English written.We called ***nltk.word_tokenize()*** function to divide strings into lists of substrings including words and punctuation. For example, the question "What are the best way to lose weight?" is tokenized to a list of (What, are, the, best, way, to, lose, weight,?) .

2. **Embedding tokens to vectors.** Here we used the Glove model developed by Jeffrey, Richard Socher, and Christopher Manning from Stanford University. They provided 4 different versions

of pre-trained word vectors and we chose the version of Twitter (2B tweets, 27B tokens, 1.2M vocab, uncased, 200d vectors) as our embedding model. In this model, each word of one question is embedded to a 200-dimension vectors and we calculated the average of all Glove vectors as the final embedding results for each question.

After embedding, every question is converted to a 200-dimension vectors so that we can calculate similarity of question pairs using vector-level methods. Here we used the Euclidean distance and Manhattan distance to quantity the similarity.

Equation for Euclidean distance calculation:

$$d(p,q) = d(q,p) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \ldots + (p_{200} - q_{200})^2} = \sqrt{\sum_{i=1}^{200} (p_i - q_i)^2}$$

Equation for Manhattan distance calculation:

$$d_1(p,q) = d_1(q,p) = |p - q|_1 = \sum_{i=1}^{200} \left|p_i - q_i\right|_1$$

Features added in this step including:
- question 1 Glove embedded vectors: ***embedding1***
- question 2 Glove embedded vectors: ***embedding2***
- Euclidean distance calculated using the GloVe question embedding: ***euc_dis_emb***
- Manhattan distance calculated using the GloVe question embedding: ***manh_dis_emb***

In general, we selected all numeric features for modelling.

## 2.4 Data Rebalance

There are 36.9% positive instance and 63.1% negative instance in our data. The two classes are unbalanced. The original plan is to rebalanced the data. However, the model that works well on the rebalanced data might not be able to generated to unbalance data. Our ultimate goal is to give a good prediction of duplicate/not duplicate for any two sentences that are input into our best model from new data. In order to solve this problem, we created a held-out testing set and separated the modeling into original data modeling part and rebalanced data modeling part in training set. In this case, we can find which model in each context gives the best results in training set and finally apply to the testing set contains original data. Details of modeling plan will be introduced in the modeling section.

## 3. Modeling

For model selection, we separated our data in the following way. First, we separated our held-out test set and train set using a 2:8 ratio. The held-out test set is always kept unbalanced in order to test for model generalization.

Then the train set is separated using two ways:

- Original training set:

- 80% of the train set: unbalanced train set, used to train the model and doing cross-validation to tune hyper-parameters for each model
- 20% of the train set: unbalanced hold out test set to compare how our models perform and select the best model to be used on the unbalanced test set we separated out in the first step.

- Rebalanced training set:
    - 80% of the train set: rebalanced train set, used to train the model and doing cross-validation to tune hyper-parameters for each model
    - 20% of the train set: rebalanced hold out test set to compare how our models perform and select the best model to be used on the unbalanced test set we separated out in the first step.

We are separating the training set into a train and test set again because we want to select the best model trained on the rebalanced/unbalanced dataset, and compare the best model of its kind on the unbalanced test dataset for generalization purpose.

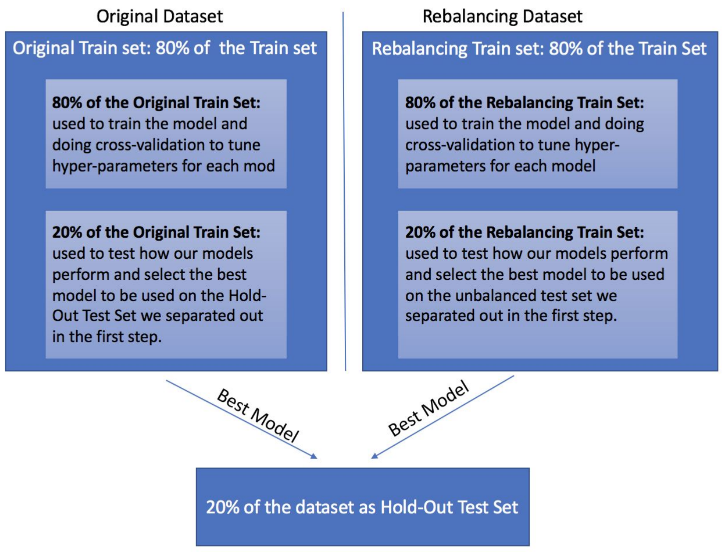

Original Dataset
Original Train set: 80% of  the Train set
80% of the Original Train Set: used to train the model and doing cross-validation to tune hyper-parameters for each mod
20% of the Original Train Set: used to test how our models perform and select the best model to be used on the Hold-Out Test Set we separated out in the first step.
Rebalancing Dataset
Rebalancing Train set: 80% of the Train Set
80% of the Rebalancing Train Set: used to train the model and doing cross-validation to tune hyper-parameters for each model
20% of the Rebalancing Train Set: used to test how our models perform and select the best model to be used on the unbalanced test set we separated out in the first step.
Best Model
Best Model
20% of the dataset as Hold-Out Test Set

Figure 4: Model Plan

## 3.1 Baseline Models

**Always Dominant class:** The simplest way to approach this binary classification problem is by aways choosing the dominant class. The distribution of the target variable in the testing set for duplicate questions and different questions are around 37% and 63% respectively. If we determine all question pairs as different, in other words, if we predict all labels to be 0, **We can achieve 12.8 log-loss and 0.5 AUC.**

**Logistic regression model using basic sentence structure features:** We also tried the logistic regression with basic sentence structure features. As we know, logistic regression is a simple model which is computationally fast. However, it emphasizes the general pattern but is not able to capture the detailed high level features and characteristics so that it sometimes cause the problem of underfitting. Also, we are using the basic sentence structure features and ignore the semantic meaning of questions. **The log-loss of basic logistic regression is 9.2 and the AUC is 0.68.**

## 3.2 K Nearest Neighbors

The first model we start with is K Nearest Neighbors. It is a simple machine learning algorithm that uses the majority votes of its nearest neighbors to categorizes an input. KNN is non-parametric and robust, thus its decision boundary can take on any form. This makes KNN a good model for a start point.

KNN has two important parameters, n neighbors, and weights. We used GridSearch with 5-fold Cross Validation from Sklearn to tune hyperparameters for the KNN model in order to achieve a better result for the model. KNN is computationally expensive on our dataset. We were only able to tune the n_neibors from a range of 1 to 10, and weight_options of 'uniform' or 'distance'. As discussed in modeling plan part, we tuned hyperparameters for both balanced and unbalanced data,and the table below shows the results.

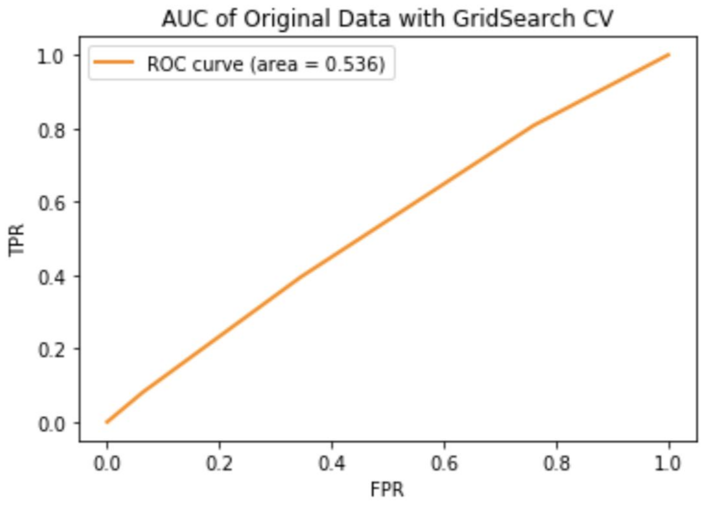


Figure 5: AUC for KNN

| **Dataset** | **Models** | **Turned Parameters** | **log-loss** | **AUC** |
|---|---|---|---|---|
| **Original data** | Base KNN | | 3.86 | 0.69 |
| | GridsearchCV | algorithm='auto', metric='minkowski', n_neighbors=3, weights='uniform' | 3.34 | 0.67 |
| **Rebalancing data** | Base KNN | | 3.86 | 0.68 |
| | GridsearchCV | algorithm='auto', metric='minkowski', n_neighbors=3, weights='uniform' | 3.34 | 0.69 |

## 3.3 Random Forest

Random forest model is an ensemble model of decision tree classifiers, which subsample the dataset and also features and train each tree classifier. RF constructs a multitude of decision trees during the training period and output the class that is the mode or average of the tree classifiers' result, thus improves accuracy and prevents over-fitting. In our case, since our data is more of distances and similarity within a small range, we do not face severe over fitting problem caused by high-cardinality categorical variables.

For the random forest model, the main parameters can be divided into two levels: 1. Forest-level parameters: n_estimators and max_features; 2. Tree_level parameters:max_depth, criterion, min_samples_split, and min_samples_leaf. We also used  the gridsearch with 5-fold cross validation to tune parameters including criterion, max_depth, max_features,n_estimators, the table below shows the results.

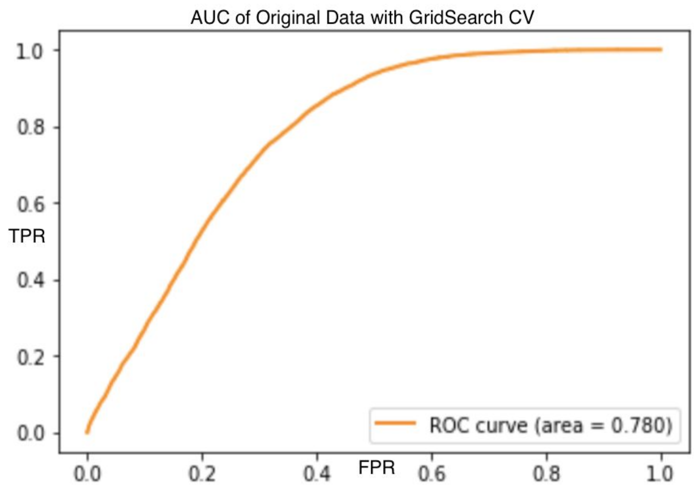


Figure 6: AUC for RF

| Dataset | Models | Turned Parameters | log-loss | AUC |
|---|---|---|---|---|
| **Original data** | Base RF | | 1.525 | 0.746 |
| | GridsearchCV | criterion: gini, max_depth: 6, max_features: auto, n_estimators: 500 | 0.519 | 0.780 |
| **Rebalancing data** | Base RF | | 1.477 | 0.729 |
| | GridsearchCV | criterion: gini, max_dept: 4, max_features: auto, n_estimators: 200 | 0.408 | 0.773 |

## 3.4 XGBoost

XGBoost, known as Extreme Gradient Boosting, is an ensemble tree method with gradient boosting framework. It has a more efficient and accurate performance by using ensemble trees model than other decision tree model as it recrifies the errors of the previous classifiers until the training data is accurately predicted by the model. The parallel and distributed computing adopted in XGBoost can expedite the model's computation speed. Hence this algorithm is adopted widely used in Kaggle and had achieved very good results in many cases.

We were able to do Grid Search with 5-fold cross validation on some booster parameters with our own laptop for original data modeling and rebalancing data modeling. We choose the learning rate to be 0.01 and number of estimators to be 700, which is roughly the square root of sample size,  and tuned tree-specific parameters includes max_depth, min_child_weight, gamma, subsample, colsample_bytree for decided learning rate and number of trees. According to different evaluation metrics, the table below shows the results.

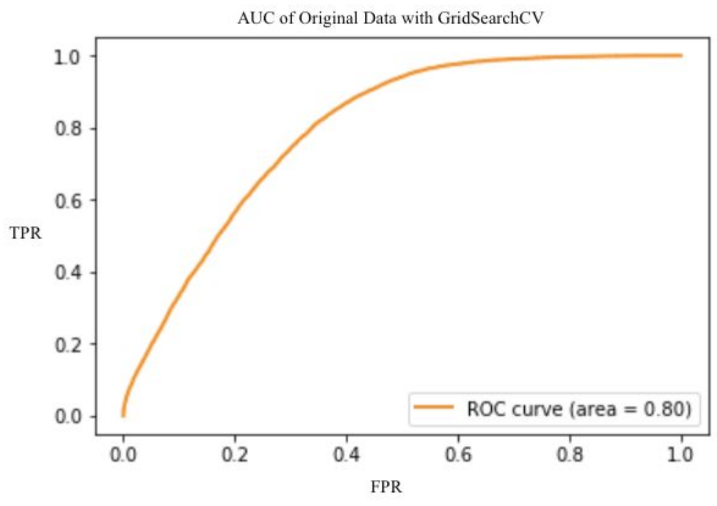


Figure 7: ACU for XGboost

| Data Set | Modeling | Tuned parameters | log-loss | AUC |
|---|---|---|---|---|
| **Original data** | Base XGboost | | 0.515 | 0.78 |
| | GridsearchCV | gamma: 0.2,<br>max_depth: 9,<br>min_child_weight: 4<br>Subsample:0.8<br>colsample_bytree:0.8 | 0.508 | 0.80 |
| **Rebalancing data** | Base Gboost with | | 0.373 | 0.78 |
| | GridsearchCV with | gamma: 0.1,<br>max_depth: 9,<br>min_child_weight:1<br>Subsample:0.8<br>colsample_bytree:0.8 | 0.371 | 0.79 |

## 4. Evaluation

### 4.1 Final Model

Now, we have the results for original data set and rebalancing data set respectively. For the original data set, the best model is the XGboost with tuned hyperparameter, which have log-loss of 0.508 and AUC of 0.8. For the rebalancing data set, the best model is also the XGboost with tuned hyperparameter, which achieve log-loss of 0.371 and AUC of 0.79. Next, we are going to test the two best models for each dataset on the 20% hold-out test set, which contains the new original data, and we get the following test results.

| | **log-loss** | **AUC** |
|---|---|---|
| **Best Model(XGboost) on Original Dataset** | 0.506 | 0.79 |
| **Best Model(XGboost) on Rebalancing Dataset** | 0.567 | 0.79 |

### 4.2 Evaluation Metrics

In regard to evaluation metrics, We used logistic loss and area under curve (AUC) to evaluate our classification prediction models.

**Log loss**

Logarithmic loss(log loss) outputs a probability value between 0 and 1. It measures the performance of a classification model. It is always compared to accuracy, which is the count of true prediction. Accuracy is a binary indicator of a yes and no. However, it is sometimes not good enough because of its binary mature. Log loss, on the other hand, outputs a probability and takes uncertainty into account. It is more informative regarding how our model performs. Small log loss indicates that the model performs well. A perfect model has a log loss of 0. When the predicted probability being 1 for an instance is small, while the true label is 1, loss log is high.

**AUC:**

Area under the curve(AUC) is another common evaluation metrics for classification model. To explain this AUC intuitively, we randomly choose two sample, one positive example and one negative example. We use the classifier to train these two example, the probability of the positive example being labeled as positive is $p_1$, and probability of the negative example being labeled as positive is $p_0$. AUC represent the probability of $p_1 > p_0$.

Comparing these two metrics, we choose log loss to select our best model. It is true that AUC is more intuitive and can serve as a good reporting metrics. However, in our results, AUC does not distinguish which model perform better on a significant scale, especially for out best model, the XGBoost model.

More importantly, compared to AUC, log loss is well calibrated and takes uncertainty into account. In our case, where the ultimate goal is to develop a automation process, the output of Log loss is more informative. The output with a probability above a certain threshold will be given to machine, and the ones that are controversial will be left for human judgement

## 4.3 Error Analysis

Even Though the accuracy is relatively high, there are still some reasonable errors, resulting from data mining process and the constraints of models. By error analysis, we could explore the non-trivial irreducible errors and improve the process of our prediction. Here we analyzed the prediction of XGBoost model with the best performance among all models.

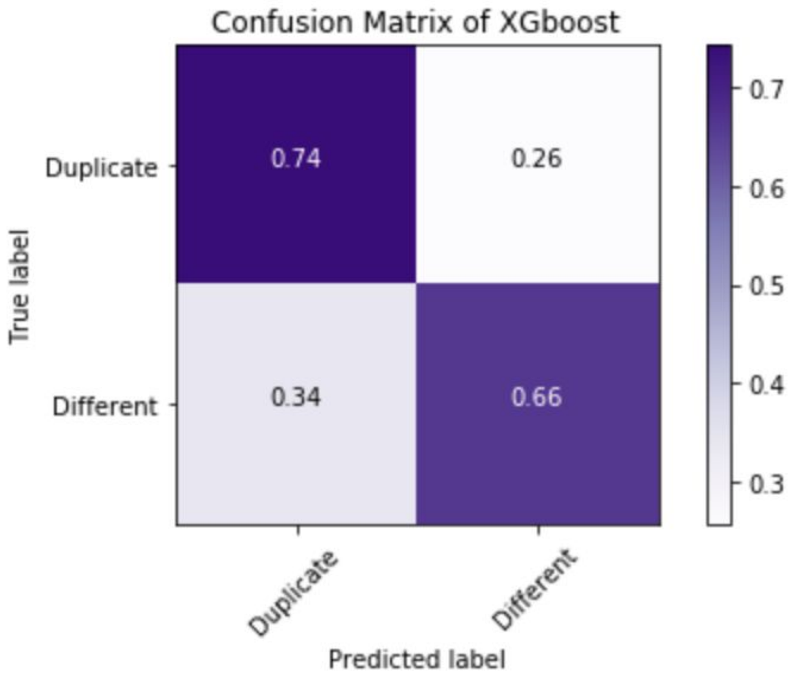


Figure 8: Confusion Matrix

We applied confusion matrix which shows the detailed Type 1 error and Type 2 error. As we can see, 74% of the duplicated question pairs and 66% of the different question paris are truly predicted. However, 26% of the question pairs has type 1 error, that is, the model predicts the duplicated question pairs as different. And 34% of the question pairs suffered from type 2 error, in other words, the model predicts the different question pairs as duplicated. Since we are focusing on the rightness of the prediction, that is, whether the model classifies the question pairs correctly, the two errors are equally severe and we want to avoid in future work.We attribute our misclassification into several error sources, which would be discussed in section 6.2.

## 5. Productization

In this informatics world, a modeling approach to detect the semantic similarity between two sentences are very important. Models that can recognize the semantic similarity between two given contents can help websites like Quora, Yahoo or StackOverflow to better organize their data by matching new questions to its solved questions. Such model would help the knowledge platform businesses to better consolidate their data, thus leading to better user experiences.  Users benefit even more from this model. If such model is successfully deployed, they no longer have to open every question to search for an answer to a question. Users will also experience less distraction when looking for solutions to a specific problem.

In this project, we added some hand-crafted feature and applied machine learning algorithms to predict a binary duplicate/not duplicate result. Our model out-performed our baseline model a lot. More specifically,  from the baseline model to our final model, the log-loss decreased from 9.2 to 0.506, and the AUC increased from 0.68 to 0.79.

### 5.1 Deployment

We developed a model to identify identical questions using machine learning approach and hand-crafted feature. The ultimate purpose of our model is to solve real-world problems and to create business value.

Models that can successfully identifying equivalent context have a huge potential market because it can be applied in many areas. Knowledge-based website, like Quora and StackOverflow can use such model to aggregate similar questions. The model can also be included in recommendation system to recommend the users to check on similar questions asked. Academia uses such model for plagiarism detection. Finance industry can use these model to aggregate similar news happened today and only see relevant ones.

The product of such model is a back end engine which will generate a probability of these two piece of text being semantically equivalent. We can let the machine determine if they are equivalent by setting a threshold for the probability generated in model results. Alternatively, human judges can look at the ones that are controversial according to the result and let the machine categorize the ones whose results we have confidence in.

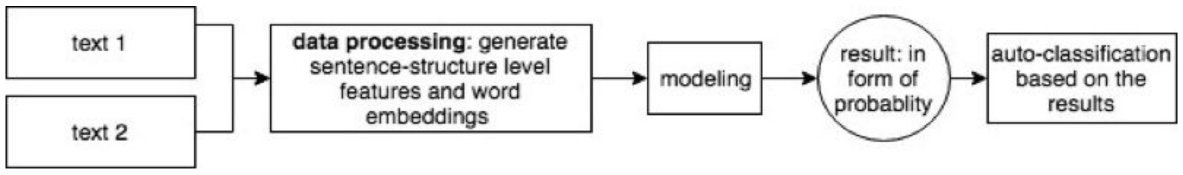

text 1
text 2
data processing: generate sentence-structure level features and word embeddings
modeling
result: in form of probablity
auto-classification based on the results

Figure 9. Workflow of a sentence

### 5.2 Limitations and potential improvement

**Insufficient Features**: Since there is no feature that can be directly used in our models from the original data set, we had to do features engineering by our experience, which is relatively subjective. In this project, we mainly used basic sentence features such as the length and tokens of sentences and used pre-trained Glove model as our embedding tool to generate vector-level features. But we didn't deal with many other latent important features such as the character length of each question without space, the difference of punctuation marks in each question pair, n-grams for each question and fuzzy-matching features, which may play an important role in similarity identification. So in the feature work, we can generate more meaningful features to improve the model performance.

**Limited Models Selection**: Our project only employed three models including KNN, Random Forest and XGBoost without any deep learning configuration. Recently, deep learning architectures and algorithms have already made impressive advances in fields of NLP. And the top 10% teams of the kaggle quora question pairs competition have applied neural networks which can perform well on the test data with around 0.116-0.138 log loss. The potential improvement we can do here is to work on some neural networks such as Siamese LSTM or ESTM [6].

## 6. Conclusion and Future step

In this report, we tried to tackle on the problem of identifying semantically equivalent sentences using machine learning models.

We faced the problem of imbalanced class. Considering that, at the time of deployment, we will be facing dataset with unknown distribution, we compared the model performance on balanced and unbalanced dataset. Surprisingly, the model trained on unbalanced, original dataset performs better. Our work proved, when facing imbalanced class, it is not always better to balance the classes. Eventually, our best model(XGBoost) achieve a log loss of the 0.506.

Constraints still exists in our model. Given more time and more resources, we could enhance the model by applicating deep learning and mining more knowledge-level features(different word embedding). Finally, our work would be expanded into a automatic system that can identify similar contexts. If successful, this system can help businesses to better organize and aggregate data, saving people more time. It can be further incorporated into a recommendation system to recommend context that one might be interested in reading.

## 7. Collaboration Statement

Starting with a clear plan, we distributed the workload equally among our group members and also helped each other a lot. Every member drafted on their data processing methods and models, and completed the final report together.

## Reference

[1] Gruen, Armin, and Devrim Akca. "Least squares 3D surface and curve matching." ISPRS Journal of Photogrammetry and Remote Sensing 59.3 (2005): 151-174.

[2] Landauer, Thomas K., Peter W. Foltz, and Darrell Laham. "An introduction to latent semantic analysis." Discourse processes25.2-3 (1998): 259-284.

[3]Pennington, Jeffrey, Richard Socher, and Christopher Manning. "Glove: Global vectors for word representation." Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP). 2014.

[4] Akca, Devrim. A new algorithm for 3D surface matching. ETH Zurich, 2004.
Landauer, Thomas K. Latent semantic analysis. John Wiley & Sons, Ltd, 2006.

[5] Chen, Qian; Zhu, Xiaodan; Ling, Zhenhua; Wei, Si; Jiang, Hui; Inkpen, Diana，  Enhanced LSTM for Natural Language Inference. 2016

[6] Mueller, Jonas, and Aditya Thyagarajan. "Siamese Recurrent Architectures for Learning Sentence Similarity." AAAI. 2016.