Kaggle Quora Pairs Report

---by Donk.Wang 2019.12

1 Definition

1.1 Project Overview

Quora is a platform to ask questions and connect people who provide unique insights and quality answers. There are multiple questions containing same intent which can cause seekers to spend more time to find best answers, and also make writers feel they need to answer multiple version of same questions. To reduce redundant questions, Quora want us to develop Machine Learning (ML) models to accurately decide whether a pair of question is duplicate [1] .

Question pairs similarity identification involves multi task of neutral language processing (NLP), which is the base of searching engine, conversation task, and text classification. Traditional NLP methods include bag of words (BOW), term frequency with inverse document frequency (tf-idf), word mover distance, statics distance based on sentence vector representation, etc. Recently, deep learning (DL) method with embedding such as Convolution Neural Network (CNN), Recurrent Neutral Network (RNN) also have good performance on many NLP tasks. In this project, I will apply both of them and investigate the factors that influence the results.

Quora provides 404290 question pairs as train dataset and 2345796 pairs as test dataset on Kaggle platform [1] . Each pair of question in train dataset is labeled 1 as they are duplicate, and 0 as not. Based on that, our goal is to predict whether each pair is duplicate in test dataset.

1.2 Problem Statement

Judging a pair of question have same intent or not is a supervised binary classification problem. This project require to submit PROBABILITY of a question pair has same meaning, and evaluate the submission by calculating LOGLOSS between prediction and ground truth.

First, I will explore training data by visualization and apply data preprocessing method to standard inputs. Then extracting useful information, such as constructing different features, calculating similarity and word embedding. After that, DL models including multiply perceptron (MLP), CNN and RNN model will be setup and tested respectively. Finally, the combination of all of them will be used to predict the test dataset.

1.3 Metrics

The metrics for this project is defined by Quora that calculating the LOGLOSS between prediction and ground truth:

$$\log loss = -\frac{1}{m}\sum_{i=1}^{m}(Y \cdot \log(P) + (1-Y) \cdot \log(1-P)) \tag{1}$$

Where m is the amount of samples, Y is sample label (0 – not duplicate, 1 – duplicate), P represent the prediction PORBABILITY the two sentences are duplicate.

Moreover, I will also consider the time cost of model prediction process as another evaluation. Because in real world, a scalable algorithm can react more rapidly and provide more time for backend analysis.

2 Analysis

2.1 Data Exploration

Let's first randomly sample 5 pairs to view. See table 1 below.

table 1. samples of training dataset

	id	qid1	qid2	question1	question2	is_duplicate
302213	302213	425213	389685	Why are answers on Quora always politically co...	Why the users of Quora are so politically corr...	1
150928	150928	237423	42715	How cann I spy on LINE without a target phone?	How do I hack/spy on someones LINE chat messages?	0
71680	71680	123303	123304	In biology, what is diffusion equilibrium?	What is diffusion in biology?	0
3292	3292	6526	6527	How hard is the AWS Developer Associate certif...	Is AWS CLI essential for AWS Sysops Admin or C...	0
129731	129731	194530	66418	Why can't you comment on answers anymore?	Why can't I comment on certain answers?	1

It seems "id", "qid1", "qid2" just represent the indices for each question pair and no use for judgment, even though one in Kaggle found "qid" contain the time information[2] . Then I will drop them and only keep two questions and their labels (same as test dataset). Besides, I also found some interesting question pairs listed in table 2 below.

table 2. some interesting question pairs in train dataset

ID	Question1	Qustion2	Is_duplicate
332561	Is there any law in India which stops companies from making female employees wait in office till very late at night?	Is there any law in India which stops companies from making female employees wait in office till very late at nigh?	0
78271	I am 17 now, how can I "earn" my first house or Lamborghini within 5 years? (One of my hobbies is Animation if that matters.)	I am 17 now, how can I "earn" my first house or Lamborghini within 5 years? (One of my hobbies is Animation if that matters.)?	0
66762	What should I do after completing BCA ?	What should I do after completing BCA?	0
72272	What are my chances of getting into an Ivy League?	What are my chances of getting into a Ivy League?	0
82430	What is good occupation for former software developers who want a slightly more social job?	What is a good occupation for former software developers who want a slightly more social job?	0
298	On what online platforms can I post ads for beer money opportunity?	What online platforms can I post ads for beer money opportunity?	0
111341	Where can I find the best hotel in Jhunjhunu?	Where can I find the best hotels in Jhunjhunu?	0
8	When do you use シ instead of し?	When do you use "&" instead of "and"?	0
404270	What is the difference between who and how?	What is the difference between "&" and "and"?	0
404289	What is like to have sex with cousin?	What is it like to have sex with your cousin?	0

We can see from table 2 that there are same intent question pairs with a little difference which are labeled to 0. These question pairs may over 70 and should be correct. Also, we can find that there some are proper noun like BCA (ID 66762) in dataset. If one question contain NBA and the other contain National Basketball Alliance, we can understand they are the same but PC will not. Last 3 pairs are very typical samples that help to understand how to preprocess data. For example, if we drop all special character, then ID 8 pairs will be the same (but actually are different). If we remove all STOP WORDS, then ID 404270 and 404289 pairs will also looks like the same.

After that I will make a summary for both train and test dataset including unique sentences and words, as well as their union and intersection (table 3).

table 3. Summary of training and test dataset

	Unique_sent	Union_sent	Intersection_sent	Unique_word	Union_word	Intersection_word	Pairs
train	537362	4789032	112162	232534	399741	131067	404290
test	4363832	4789032	112162	298274	399741	131067	2345796

We can see from table 3 that train dataset have around 500k pairs but test dataset have over 4 million pairs. Their intersection is just 100k pairs which takes a small piece of total. Same phenomenon also occur in their vocabulary. So I guess there must be a GAP between train and test data distribution. It also indicates that we should be careful in extracting features on train dataset to avoid over-fitting.

2.2 Data Visualization

2.2.1 Label distribution

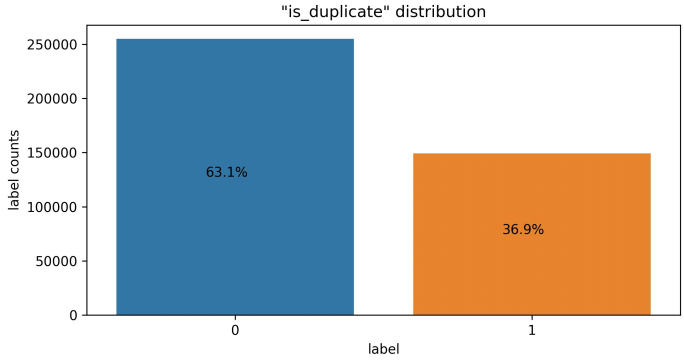


figure 1. label counts distribution in train dataset

Figure 1 shows that the ratio of “0” and “1” label is 63:37 in train dataset. It means we should consider imbalance of data and apply some technique to reduce its influence.

2.2.2 Length and word distribution

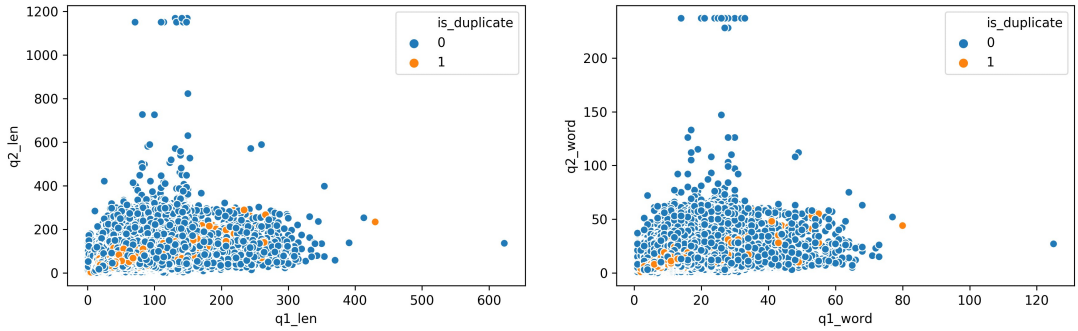


figure 2. Length and word distribution in train dataset

Figure 2 stats the lengths and words of two questions with different labels. It can be seen from figure 2 that if two question have same intent, then their lengths or words count will also be similar. We can also find there are some outliers that non-duplicate in figure, which have large gap between q1_len (q1_word) and q2_len (q2_word). The words in 99% of dataset are no more than 31, where the longest sentence have 237 words.

2.2.3 Share words distribution

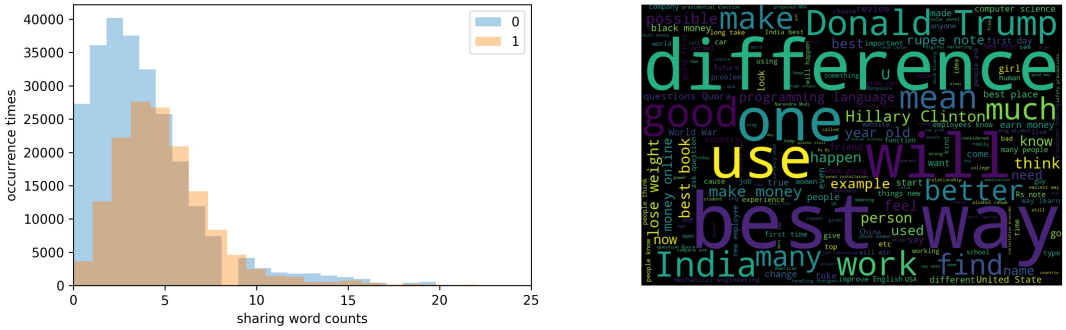


figure 3. Share words distribution
(left-share words counts distribution with different labels, right-word cloud of most popular)

The left in figure 3 shows the sharing words distribution of pair questions with duplicate and not. It seems that if sharing word counts lower than 3, they are more likely to be non-duplicate. But more sharing words do not predict they are more likely to be same. It is interesting that the right in figure 3 shows the most popular words people use. People ask about difference between two things, ask about best ways to do something, ask about Donald Trump and Hillary Clinton…

2.2.4 Text occurrence frequency

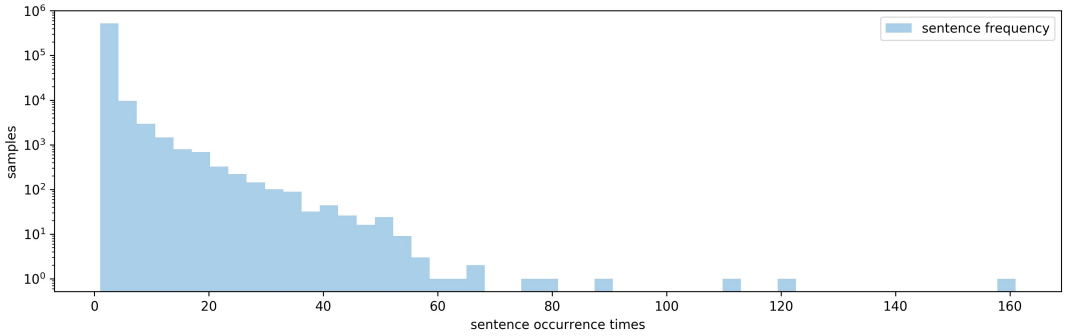


figure 4. Text occurrence frequency distribution in train dataset

The train dataset have over 400k question pairs but have only around 537k unique sentence, which means there are many questions have been used repeatedly. Therefore, I stat the occurrence distribution of train question. As shown in figure 4, the distribution of sentence occurrence is diminishing rather than randomly. Maybe this have some relations about how Quora generate those. In train dataset, the most frequent questions are “What are the best ways to lose weight”, which has been used 161 times.

2.3 Algorithm and Technique

2.3.1 Preprocessing

First I will clean data and extract features from them. Besides statics features, I will use bag of words (BoW), term frequency – inverse document frequency (tf-idf) and word embedding to vectorize sentences in word level, as the character level vectorizing is not so helpful [3] , and phrase level will generate data that takes too much spaces.

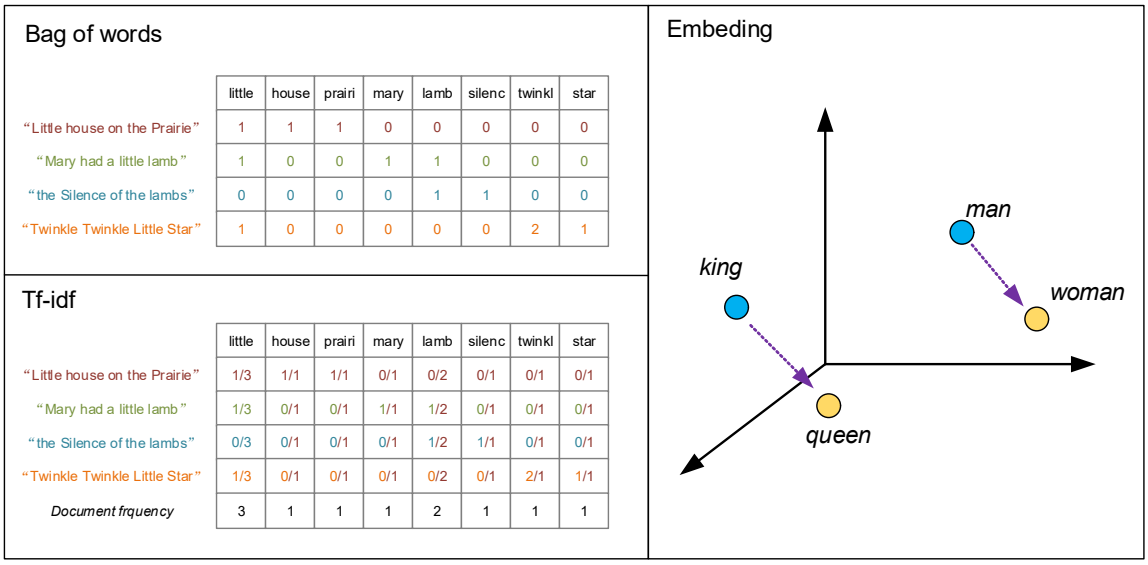


figure 5. Simplified illustration of BoW, Tf-idf and Embedding

Figure 5 gives a simplified illustration of three method. Bag of Words is a simplifying representation used in information retrieval [4] . It first collect all unique words into a "bag", then translate sentence by this "bag". If a word appear many times in a sentence (like Twinkle in figure 5), then it will be seen as important. BoW can transform a sentence to vector, but treat all words as the same. Based on idea of BoW, tf-idf consider about "inverse document frequency" [5] . This means if a word (like "is") appear many times in whole document, it more likely to be a common word and not too important. For example, "little" has been appear 3 times in 4 document, so it take little weight (1/3), where "house" appear only once and take more weights (1). In this project, I will use *sklearn* package to calculate BoW and Tf-idf vector of sentence. *Min_df* is set to 10 to avoid overfitting, and *max_df* is set to 0.9 as to reduce influence of STOP WORDS. Besides, Singular Value Decomposition (SVD) is also applied to reduce the data spaces[8] .

Word Embedding is a mapping method that convert word to vector[6] . At the same time those vectors reflect relations between words in a certain degree, such as "king-> queen≈man->woman" in figure 5. Embedding model is trained by continues bag of words (CBOW) and skip-gram []. Because training will cost much time and need large samples and calculation resource, I will use pre-trained embedding models. There are 3 popular embedding model provided by Kaggle [], Google's word2vec embedding, Glove word vectors and Facebook's FastText embedding. I have built a simple model to test their performance respectively and find Google's word2vec model is not as good as others. Moreover, Google's model and Glove model has an out-of-vocabulary (OOV) problem, where FastText model can solve in some degree [7] . Finally, I decide to use FastText model in this project.

2.3.2 Logistic Regression

Logistic Regression (LR) is designed to solve binary classification problem. It uses Sigmoid function to predict probability of label 1, which is show below:

$$y_p = sigmoid(x) = \frac{1}{1+e^{-x}} \tag{2}$$

y_p is the probability of label 1, x is the input range in (-∞, +∞). We can see Sigmoid map the data into range(0,1). If y_p larger than 0.5 then label it 1. LR model use the logloss as its cost function default, which is same with Quora's evaluation method (see equation 1). So I will use LR as BASE LINE first, then it will also be set to be the last layer for other DL model.

2.3.3 Neural Network with Siamese Architecture

Siamese neural network is first proposed to verify the signatures and then extended to be applied successful in face recognition and other matching tasks[9] [10] . It uses two artificial neural networks **sharing the same weights** and reduce the training parameters. Figure 6 shows the architecture illustration I will use in this project.

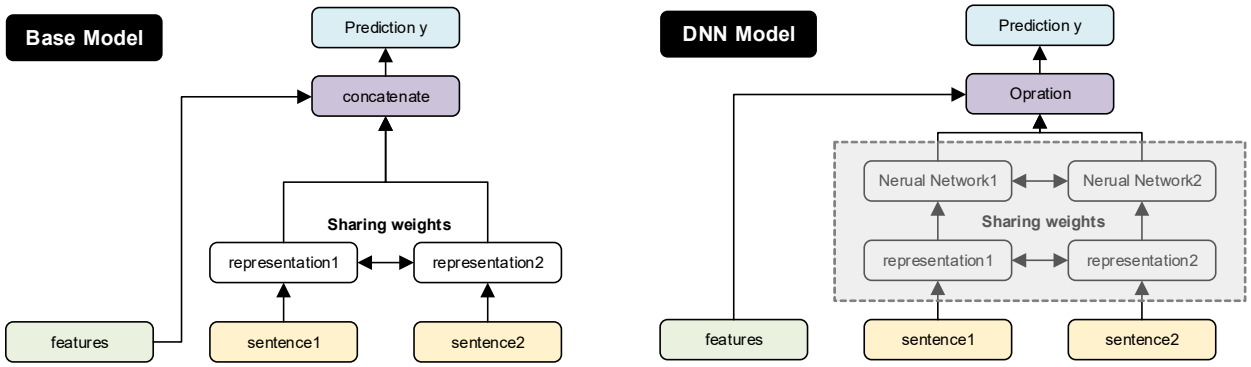


figure 6. Siamese architecture neural network in this project

As shown in figure 6, two sentences will first be embedded into two representation vector, and then they will be feed to two neural network with SAME WEIGHTS to calculate semantic vectors. After operation with features, the output vectors will be feed into last layer with sigmoid activation function. As a comparison, Base Model (left) is applied without neural network layers. For the DNN Model, neural network layer can be chosen to CNN or RNN (GRU, LSTM), which have different advantages:

- 1D CNN recently demonstrate a superior performance on those application which have a limited labeled data and high signal variations acquired from different sources (i.e., patient ECG)[11] . It can capture structure information of vectors. There are two CNN structures that commonly used in semantic matching tasks [12] , ARC-I and ARC-II.

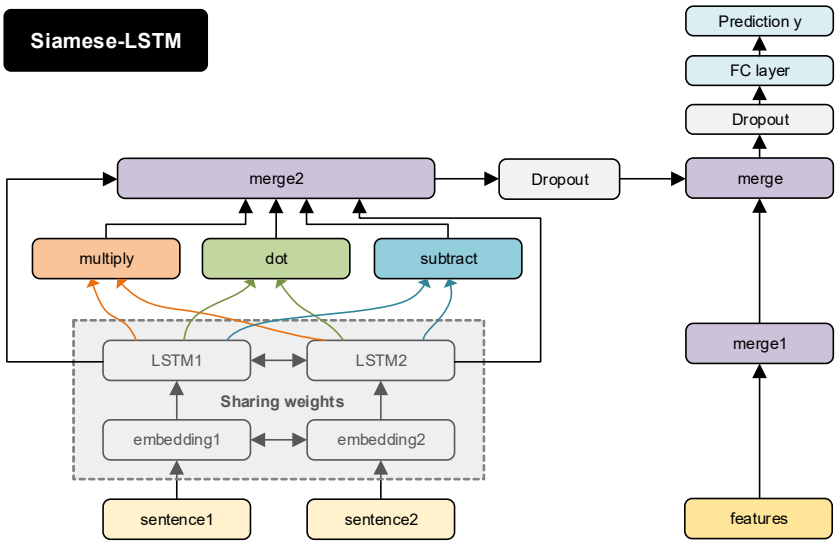


figure 7. Siamese-LSTM model illustration

- RNN (GRU/LSTM) is well known due to their capability of processing not only signal data but also entire sequence[13] . Figure 7 shows the DNN model I will use in this project. First two sentences are embedded as two representation and will be feed to LSTM. To extract information from different aspects, multiply, dot and subtract have been used on two LSTM outputs. Finally, all of them will be concatenated and feed into full connection (FC) layer with 1 sigmoid at last.

2.3.4 Post processing

In general, distribution of train dataset is different from test dataset, which may cause error in prediction, especially for logloss. In this competition, we can get the true label distribution of test dataset by submit a constant value, which is shown as follows:

First, submit your prediction with the same value, let's say P_1, and you can get the score, S_0. Assume that test dataset have R_1 (ratio) of label 1, and R_0 for label 0; then R_1 and R_0 can calculated by equation below:

$$\begin{cases} R_1 * \ln(P_1) + R_0 * \ln(1 - P_1) = -S_0 \\ R_1 + R_0 = 1 \end{cases} \tag{3}$$

I submit the 0.369 as P_1, and get S_0, 0.554. So the true ratio of label 1 in test dataset can be derived to 0.1744. Then in the training process, *class_weights* parameters are used and set as: {1:0.1744/0.369, 0:(1-0.1744)/(1-0.369)}, which does improve my scores.

2.4 Benchmark

According to Udacity's graducation requirement, my best score should up to top20% (660th/3307) in the private lead board, which has logloss of 0.18267. Besides, I also require my model is robust and not over-fitting, which means that the difference logloss between validation and test should not over 0.2. Moreover, time consuming is also an important indicator in real world and also should be considered in this project.

3 Methodology

3.1 Data Processing

Due to Internet words are non-standard, it may appear typos, capital letter, proper noun, kinds of punctuation, and abbreviation that not in dictionary. So first we need to clean them. The principle of data clean is that we should keep the data information as much as possible. In this procedure I will APPLY data clean method as follows:

- Grammar uniform: what's -> what is, we're -> we are, I'd -> I would …
- Ordinary number to number: 1'st/1st -> 1, 2'nd/2nd -> 2,…
- Split character by space: what? -> what ?, 2+1 -> 2 + 1 …
- Keep some symbols and drop the other: keep %,^,&,*,-,+…, drop <>{}[]":;?!...
- Drop redundant spaces: I have -> I have, …

Besides, I have also tried to lowercase, lemmatizing, stemming and drop stop words on sentences but have found some issues:

- Lowercase: "What is it?" and "What is IT?" are totally SAME if lowercase them, but they are actual different.
- Lemmatizing: it can decrease vocabulary size but it may make mistakes in some situation like "was" to "wa".
- Stemming: many stemmed words are not a "word" in dictionary, which cause problems in embedding process.
- Stop words: removing stop words can decrease the data size. But it may loss information like "I am happy" and "I am not happy" are the SAME after stop words removal.

Therefore, to avoid information loss, I will NOT use them during data clean procedure. Also, I will correct labels of some pairs that discussed in 2.1. Based on those clean data, I will try extract features listed bellows table 4. All these features are classified to 4 types, DIST for distance, BASE for base feature, TAG for word property and VEC for sentence representation.

Another important process is **DATA SPLIT**. Note that we already know the label distribution in test dataset (this is not work in real world), we should keep validation data with the same, 0.1744 ratio for label 1.

table 4. Input features of model

Feature	Description	Feature	Description
base feature	length, vocab and character counts of question1&2 and their difference; word & character sharing counts.	**magic feature**	Occurrence frequency of two questions; intersection and union counts of question neighbors
fuzzy features	Qratio, Wratio,...calculated by *FuzzyWuzzy* package	**Tf-idf feature**	Tf-idf vector of two questions,
sentiment	Polarity, subject degree of each question and their difference calculated by *TextBlob* package	**BoW feature**	BoW vector based on difference of question words and their word property.
distance feature	Cosine, Euclidean, Cityblock, Chebyshev,...distance calculated based on embeddings and tf-idf	**embedding**	embedding matrix by FastText with 300 dimension

3.2 Implement

Before implement, the ***input features, model configuration as optimizer (learning rate), early stopping parameters, learning rate decay strategy*** and ***random seed*** for all models should be the same. I use Adam with 0.001 learning rate for optimizer, use early stopping with 10 patience, use redcue_lr to apply learning rate decay strategy with patience=1 and decay factor=0.95. The random seed is set to 2019, and modelcheckpoint is used to save the model best performance on validation dataset. Only 5% of data is used to be validation. All experiments are run in Google Colab[14] with free GPU!

3.2.1 Base Model

Firstly, I apply the Base Model (figure 6) to calculate the results with two version, embedding and no embedding. In no embedding version, I simply use FC layer with 50 of ReLU activation as first layer. Then merge these 4 types' features and feed them into another FC with 150 of ReLU. Last they are put in 1 perceptron with Sigmoid activation layer to prediction. Experiment results are shown in figure 8.

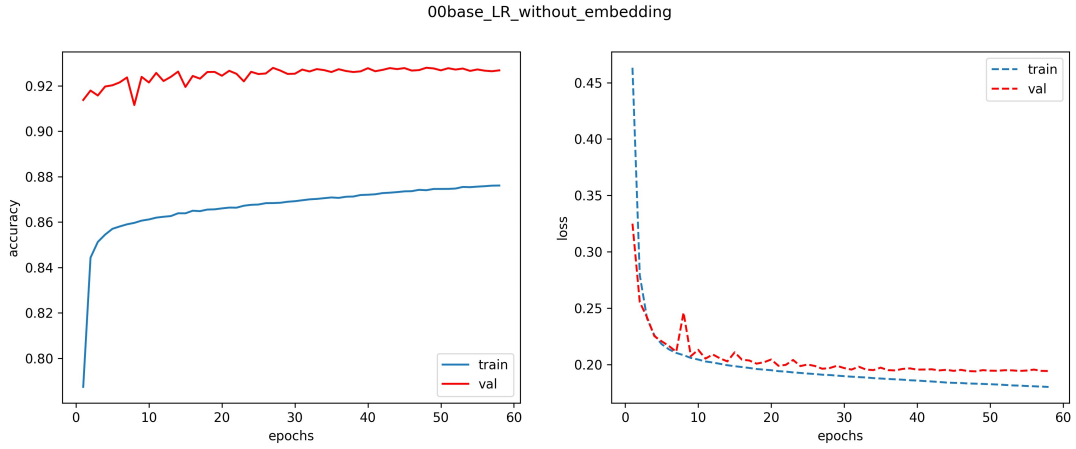


figure 8. Base Model results without embedding

The Base Model converge fast in first 10 epochs and then become slowly, but also with 3 seconds per epoch. We can see from loss curve that this model has good bias but a little variance after 40 epochs. The best val_loss is 0.1939 with val_acc 0.9280.

Base Model with embedding use the embedding represent for two questions. For a embedding vector with (pad, embedding) dimension, I use *keras.backend.sum()* to calculate the sum values along last axis, and get (embedding,) vector. Then I merge two representation vector with other features into the rest FC layer as no embedding model does. Figure 9 show that with embedding feature, variance has been reduced obviously. The best val_loss is 0.1875 with val_acc 0.9304. It can be shown that embedding is a good way to improve model performance.

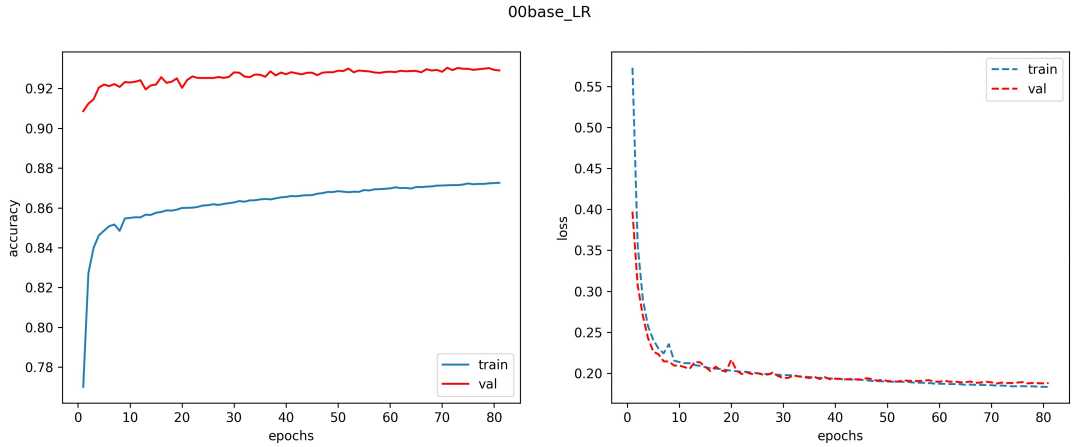


figure 9. Base Model results with embedding

3.2.2 DNN Model

I have tried DNN model with Siamese-LSTM, Siamese-BiLSTM and CNN (ARC-I, ARC-II) respectively. Due to paper limitation I will not show all the details of them but take Siamese-LSTM as an example. As shown in figure 7, initially I do not consider dropout and run the model directly. Figure 10 shows the results that training curve will never be slow which means model has been strongly over fitting. It can also be found that accuracy is higher than base model, although their losses are near. It mean two model do learn the different things, but deep model have higher variance. The best val_loss is 0.1881 with val_acc 0.9361, which better than base model without embedding but lower than base model with embedding. Next I will do some optimization on DNN model.

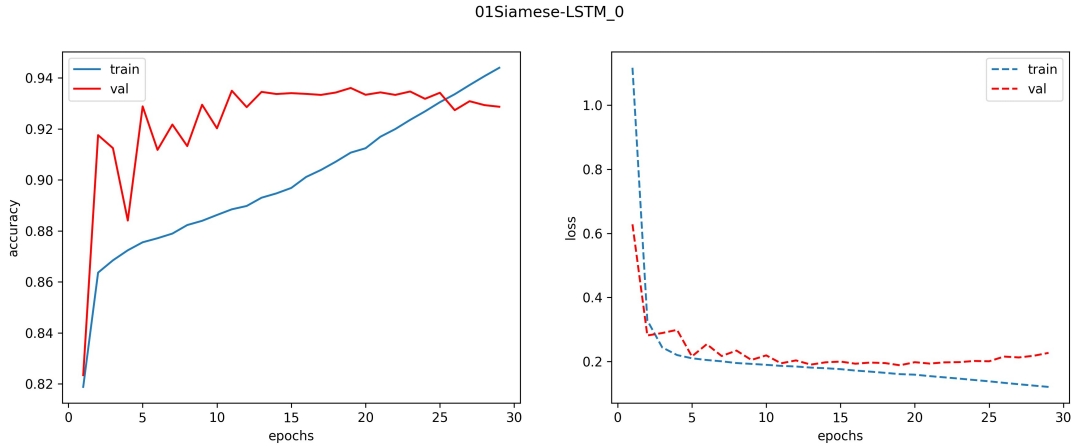


figure 10. Siamese-LSTM results without any over-fitting suppression

3.3 Refinement

There are three methods to avoid over-fitting and improve model performance: dropout, regularization and data augmentation.

- Dropout: Dropout is used to randomly set some neural activation to 0 between two layers in each training epoch, but not work in prediction. This technique is always used in DNN model to help avoid over fitting. Until now there is no consensus formula about how to add and set dropout. So I will try different dropout rate on DNN model.
- Regularization: this technique is used to suppress model weights overlarge which can also help avoid over-fitting[15] . There are two common regular method, L1 and L2. L1 regularization make some insignificant

feature weights to close to 0, where L2 regularization just decay them. L1 regularization is more suitable for spares model[16] so I will only use L2 regularization in this project.

- Data augmentation: data augmentation is the most direct way to improve the model generalization. The simple way in this project is to exchange two question position and generate pairs with same question. But we should be careful that if we exchange the position, let's say, A&B -> B&A. then in data split process, they should be together, both in validation dataset or not, if not it will make model more over fitting. But in this case, due to calculation limitation, I will not use it.

Besides the over-fitting suppression techniques, there are also many hyper parameters in model should be tuned carefully, such as learning rate, number of neural, different initialization and different architecture…. In this project, I will only try different learning rate.

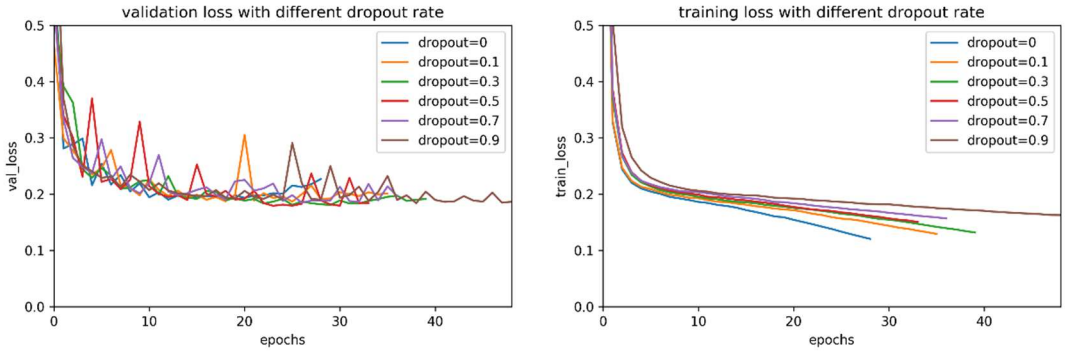


figure 11. Siamese-LSTM results with different dropout rate

Figure 11 shows the results under the different dropout rates on Merge2. The val_loss first reduce in (0~0.5) and get best score at dropout=0.5 (0.1792), then start to rise. Except blue line, the other lines are all pulled low again when they tend to go up. We can also found from training loss curve that speed of convergence become slower with larger dropout rate. There is a tradeoff between performance and time cost, which is what we should pay attention to. In this case, the suitable range of dropout rate is around 0.5.

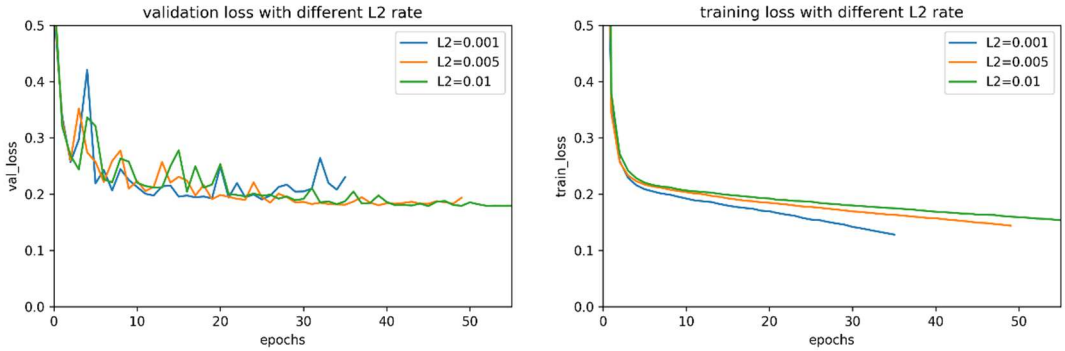
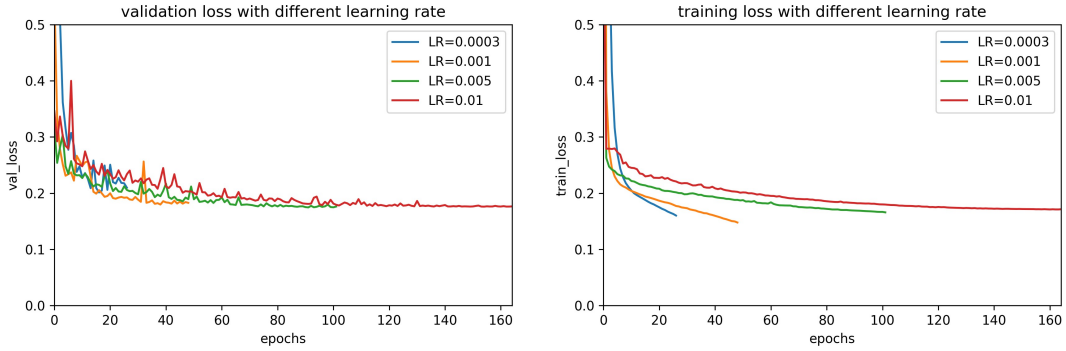


figure 12. Siamese-LSTM results with different L2 regularization rate

Similarly, I have tried three L2 regularization rate on the model. As shown in figure 12, variance of val_loss become lower with higher L2 regularization rate. It is validated that both regularization and dropout play a role in over-fitting suppression, but the difference is, higher regularization will make convergence smother while higher dropout rate will not. In this case, the best L2 regularization rate is around 0.01 (regularization rates higher than 0.01 take too much time so they are not suitable).

Last, I try kinds of learning rate on the model. Interestingly, as the learning rate (LR) rise, the convergence become slower. This may because that I apply the LR decay strategy. At first 40 epochs, the model with LR=0.01 perform worse than other 3 models as I expect. But under the LR decay, all models val_loss will converge in the same level. It means the LR decay is a good way to avoid inappropriate learning rate choice, which improve the

model performance. For example, if the best LR is 0.001 but you choice 0.01, it may take more time to converge but results will not differ too much.



4 Results

4.1 Model Evaluation and Validation

Finally, all model results are shown in table 5. Except Siamese-LSTM, other 3 DNN models have not been optimized. The optimized Siamese-LSTM achieve the best LB score, **0.15672**, which actually lower than its val_loss 0.1707. Besides, I summarize some conclusion from table 5.

table 5. Final results of all models

Model	Train_acc	Train_loss	Val_acc	Val_loss	**LB_score**	Train_time	Predict_time
Base_LR (no embedding)	0.8740	0.1830	0.9280	0.1950	0.17459	3.5min	5s
Base_LR	0.8714	0.1849	0.9304	0.1875	0.16837	7.1min	13s
Siamese-LSTM(optimized)	0.9021	0.1606	0.9395	0.1707	**0.15672**	37.7min	84s
Siamese-BiLSTM	0.9065	0.1523	0.9374	0.1759	0.16177	121.0min	182s
Siamese-CNN_ARC-I	0.8854	0.1703	0.9303	0.1867	0.17054	45.1min	26s
Siamese-CNN_ARC-II	0.8739	0.1848	0.9304	0.1881	0.18381	38.2min	74s

- **Validation reflect the test data correctly**. We can see from Val_loss and LB_score columns that their difference with one model is no more than 0.2, which meet the benchmark requirement.
- **Deep model not always beat the simple one**. It is amazing that the Base_LR model get the 0.16837 LB score that higher than two CNN model, and also takes the least time on training and predicting. This shows the good generalize capability of Logistic Regression and also may because the data feature in this project is more tend to be linear separable.
- **Time consuming is important**. Although LB score of Siamese-BiLSTM is a little better than Base_LR, the former one spend 121min to train while the latter one just need 7min. This makes BiLSTM model hard to optimize and not suitable in real world. In addition, we can also find Siamese-LSTM takes longer time to predict than Siamese-CNN_ARC-II with almost same training time.

4.2 Justification

As shown in table 5, all models except Siamese-CNN_ARC-II have achieved the LB scores better than benchmark, 0.18267. And all models are strong enough that their val_loss and LB score will no more than 0.2. But if considering time consuming, I will drop the Siamese-BiLSTM model for its long time cost. This is why I do not want to optimize it. Finally, I will use Siamese-LSTM as my best model (0.15672 LB), and also recommend Base_LR (0.16837 LB) for its high speed and generalization.

5 Conclusion

5.1 Free-Form Visualization

In this project, an important technique is class balance in post processing. Figure 13 shows the difference distribution of the same model prediction. As description in 2.3.4, the label 1 in test data in much less than in training data. So with class balance, predictions are more skew to label 0, comparing those without balance. The val_loss of left graph is 0.1707 with 0.15672 LB score, while val_loss in right graph is 0.2426 with 0.18528 LB score. Obviously, class balance help improve the LB score. It also shows that validation choice is significant for model training.

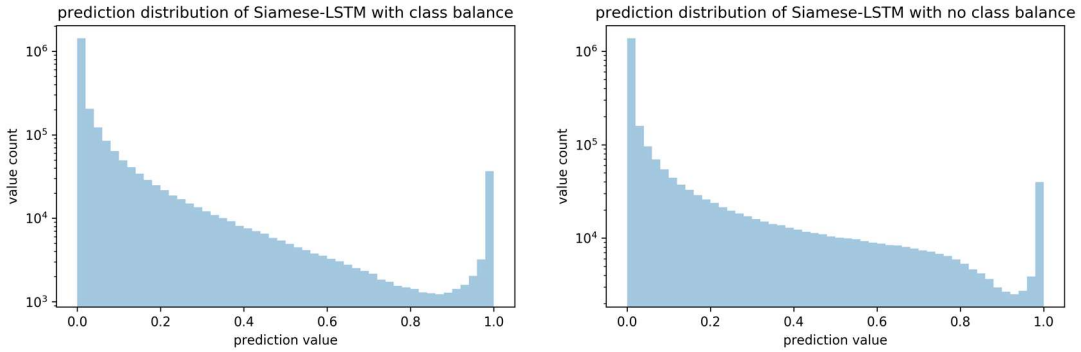


figure 13. Siamese-LSTM prediction distribution(left-class balance, right-no class balance)

Another amazing thing is the magic feature. I just remove the magic feature for optimized Siamese-LSTM, and found the model turn to be worse significantly, which is shown in table 6. Magic feature importance on Siamese-LSTM(optimized) performance

features	Train_acc	Train_loss	Val_acc	Val_loss	**LB_score**	Train_time	Predict_time
Magic	0.9021	0.1606	0.9395	0.1707	0.15672	37.7min	84s
No Magic	0.9079	0.1705	0.9099	0.2461	**0.29205**	54.9min	83s

At beginning of this project, I first build kinds of models and extract all features except magic as I can, but their LB score will always not less than 0.3. I search the reason and others' jobs but also find they also could not surpass 0.3. Later I find the magic feature in the discussion forum and finish the goal. Quora said there are many question pairs they using some technique to generate. So I think the top models are just over fit the "model" Quora used. **This is the most different feature between competition and real world**.

5.2 Reflection

Here I want to share 3 points that trouble me a lot in beginning.

1. **Be patient and explore data sufficiently**

   At first I simply generate some base features and using Google word2vec embedding without data exploration. This gives me around 0.4 LB score. To achieve better scores quickly, I change features and model structure at the same time, which make me confused. Then I fall into the cycle of feature extraction and model training, which takes me around one month but do not achieve improvement. A better way is that be patient on feature engineering and do not eager to build complex model. Start with a simple model and sample your data that help you build up a pip-line quickly. While the engineering mature, then you can try more advanced model. This will save your time.

2. **Extra information is the best way to improve your score**

Because this project is a competition rather than real world problem, we need to utilize the test dataset information as much as possible. For example, you can apply standardize transform on **union** of train and test data rather than train data only. Moreover, you need to sufficiently extract features especially for test data "leakage" (magic), which will help improve your score drastically. In this project, text occurrence frequency is one of those "leakage" features.

3. **Concentrate on your target**

   The evaluation of project is logloss rather than accuracy. Note that a results with 0.8 accuracy and 0.2 logloss **is better** than one with 0.85 accuracy and 0.25 logloss. Therefore, we should improve our logloss rather than accuracy, which require our validation is nearer with data distribution. In this case, the key point is figure out the test data label distribution and keep the validation with the same ratio.

5.3 Improvement

I think that there are 3 aspects can help improve my LB score.

1. **Models**: As I said in 4.1, except Siamese-LSTM, the other DNN model has not been optimized which have an improvement space. It is found that one in kaggle using CNN even get 0.14 LB score[17] . furthermore, I believe attention mechanism can also help improve my RNN model performance.
2. **Features**: there are still many latent useful information that I do not grab such as word mover distance.
3. **Ensemble**: this technique will certainly improve your LB score in any competition, and I will also learn to use it in next project.

6 Reference

[1] https://www.kaggle.com/c/quora-question-pairs/overview

[2] https://www.kaggle.com/c/quora-question-pairs/discussion/34335#latest-191013

[3] https://www.kaggle.com/c/quora-question-pairs/discussion/34355#latest-674196

[4] https://en.wikipedia.org/wiki/Bag-of-words_model

[5] https://en.wikipedia.org/wiki/Tf%E2%80%93idf

[6] https://en.wikipedia.org/wiki/Embedding

[7] https://cloud.tencent.com/developer/article/1434913

[8] https://zh.wikipedia.org/zh-hans/%E5%A5%87%E5%BC%82%E5%80%BC%E5%88%86%E8%A7%A3

[9] Bromley, Jane, et al. "Signature verification using a" siamese" time delay neural network." Advances in neural information processing systems. 1994.

[10] Nair, Vinod, and Geoffrey E. Hinton. "Rectified linear units improve restricted boltzmann machines." Proceedings of the 27th international conference on machine learning (ICML-10). 2010.

[11] Kiranyaz, Serkan, et al. "1D Convolutional Neural Networks and Applications: A Survey." arXiv preprint arXiv:1905.03554 (2019).

[12] Hu, Baotian, et al. "Convolutional neural network architectures for matching natural language sentences." *Advances in neural information processing systems*. 2014.

[13] https://en.wikipedia.org/wiki/Long_short-term_memory

[14] https://colab.research.google.com/notebooks/welcome.ipynb

[15] https://towardsdatascience.com/l1-and-l2-regularization-methods-ce25e7fc831c

[16] https://stats.stackexchange.com/questions/45643/why-l1-norm-for-sparse-models

[17] https://www.kaggle.com/rethfro/1d-cnn-single-model-score-0-14-0-16-or-0-23