CISC 6210
Natural Language Processing
Prof. Yanjun Li
Dec.18, 2019
Final Project

Quora Duplicate Questions Detection

Jie Lu

Abstract

It is generally considered challenging to decide whether two questions are intended to ask the same content, because there are a variety of choices of words and sentence structures. I used the dataset launched by Quora for this task, the well-known question-and-answer platform. The dataset consists of 400K+ labelled question pairs. I referred to the architecture of the academia benchmark - Siamese LSTM, while implementing my own LSTM structure to tackle this problem, and reached a competitive score. A major focus of this project is on the hyperparameter tuning. I tuned some major hyperparameters, made a thorough comparison and yielded the best available model under my experimental setting. Moreover, the deep learning techniques significantly outperform traditional statistical learning baseline.

Introduction and Background

Quora , as one of the most famous Q&A websites, has reached over 300 million active users per month. It provides users with a platform to ask questions that other users on the site may answer. However, there are increasingly more questions being asked today that were already asked beforehand, often with different wording and phrasing. Therefore, ideally, these duplicate questions would be merged together into a single question. For this purpose, Quora launched its duplicate question detection Kaggle challenge in the mid 2016, with real question data collected over 2015 - 2016. Solving this problem would be beneficial in the following aspects

- It saves question seekers' time to get answers to their questions. They no longer need to poster questions that are existent and wait there for someone to answer them.
- It saves question answers' time that they don't have to copy and paste their same version of the answers repeatedly to duplicate questions.
- Seeing duplicate questions very frequently would give a negative impact on user experience. Removing duplicates would improve the overall content quality and increase user engagement in the long run.
- It clears the redundancy of the database.
- Knowing the patterns of alternative word choosing and representation can improve Quora's searching query quality. As a Q&A platform, recording those patterns would be helpful to build its own customized NLP system and text corpus.

Data Description

id	qid1	qid2	question1	question2	is_duplicate
447	895	896	What are natural numbers?	What is a least natural number?	0
1518	3037	3038	Which pizzas are the most popularly ordered pizzas on Domino's menu?	How many calories does a Dominos pizza have?	0
3272	6542	6543	How do you start a bakery?	How can one start a bakery business?	1
3362	6722	6723	Should I learn python or Java first?	If I had to choose between learning Java and Python, what should I choose to learn first?	1

Figure 1: data sample

The original data shape is (40290, 6), accounts for 60MB. As an unstructured text data, it's relatively clean, as all data is real questions on Quora collected between 2015 and 2016. There is no missing value, and each row pair is unique.

It's worth attention that by saying 'duplicate', Quora is interested in finding out question pairs that expressed the same intention. Some questions can be labelled as duplicate easily such as "How long does it take to fly from New York to Shanghai?" v.s. "How many minutes would it need to fly from New York to Shanghai?". However, some pairs can be tricky and almost impossible to determine whether it's duplicate. For example, it's impossible to differ "How do I make 100K bucks fast?" from "How do I make 100K dollars fast?". It's only duplicate when you believe buck is equal to dollar. Therefore, it's natural that the labels contain some nose, as all the labels are done by human experts.

Exploratory Data Analysis and Visualization

Before approaching the modelling phase, I conducted several exploratory analysis to better understand the data. First, I noticed the labels are quite balanced, and duplicate labels account for 37% of the total data. For this task, I would not oversample or undersample the data, but when it comes to the situation that the underlying distribution of the test data is different, I would reweight or resample the training data.

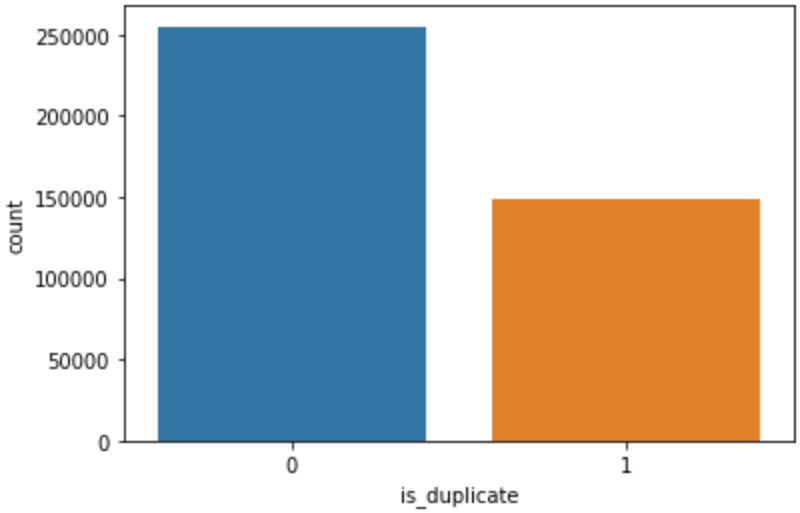


Figure 2: label distribution

I also plotted the log-histogram of question occurrence, the distribution of character and words. All of them are positively skewed. Most questions appear only once, and there are several outliers appearing more than 100 times. For character distribution, there is a small cluster at 150, because at that time, 150 is the maximum character limits.

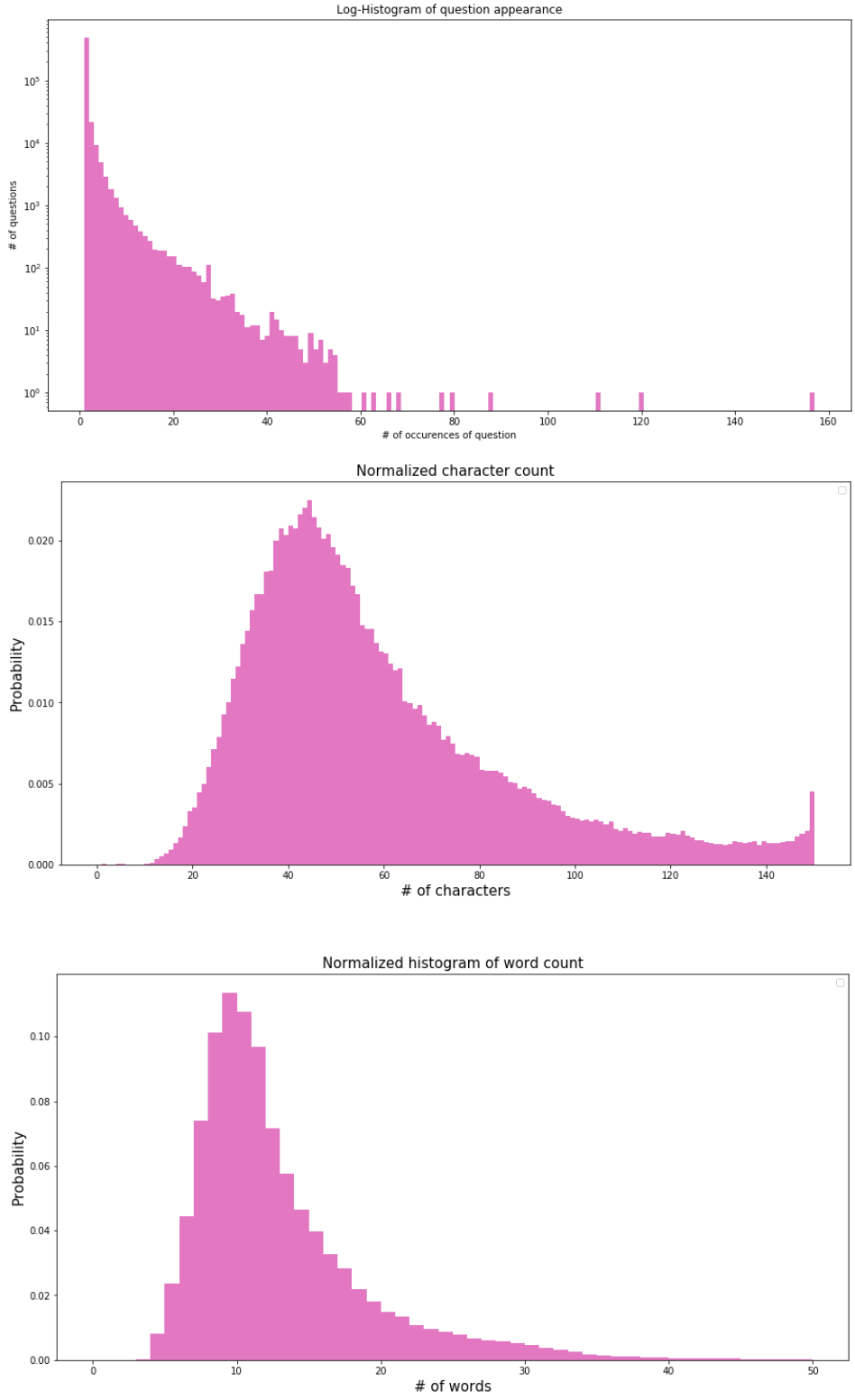
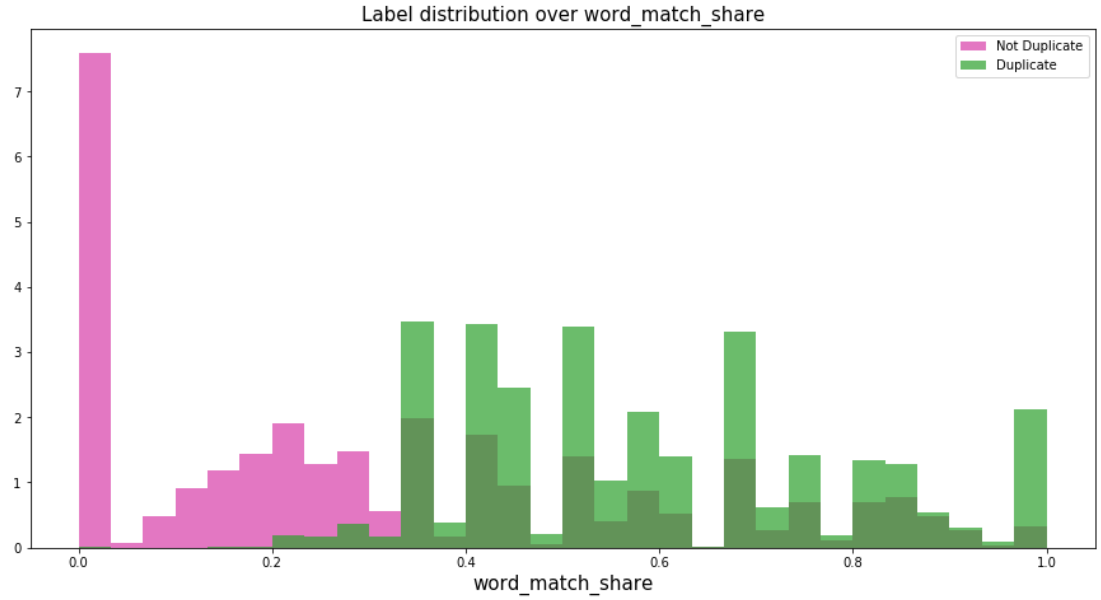


Figure 3: question occurrence, normalized character counts and normalized word counts

The word share feature is created to check what percentage of the words are shared between the question pairs. It turns out that the word share can be a good feature to separate the non-duplicate question pairs, because they have a quite small amount word shares. The violin plot is simply tackling the same relation with a neater view. We can see the non-duplicate questions are fatter at the bottom, while the duplicate questions are fatter in the middle and upper part of the body, that makes it hard to be separated.



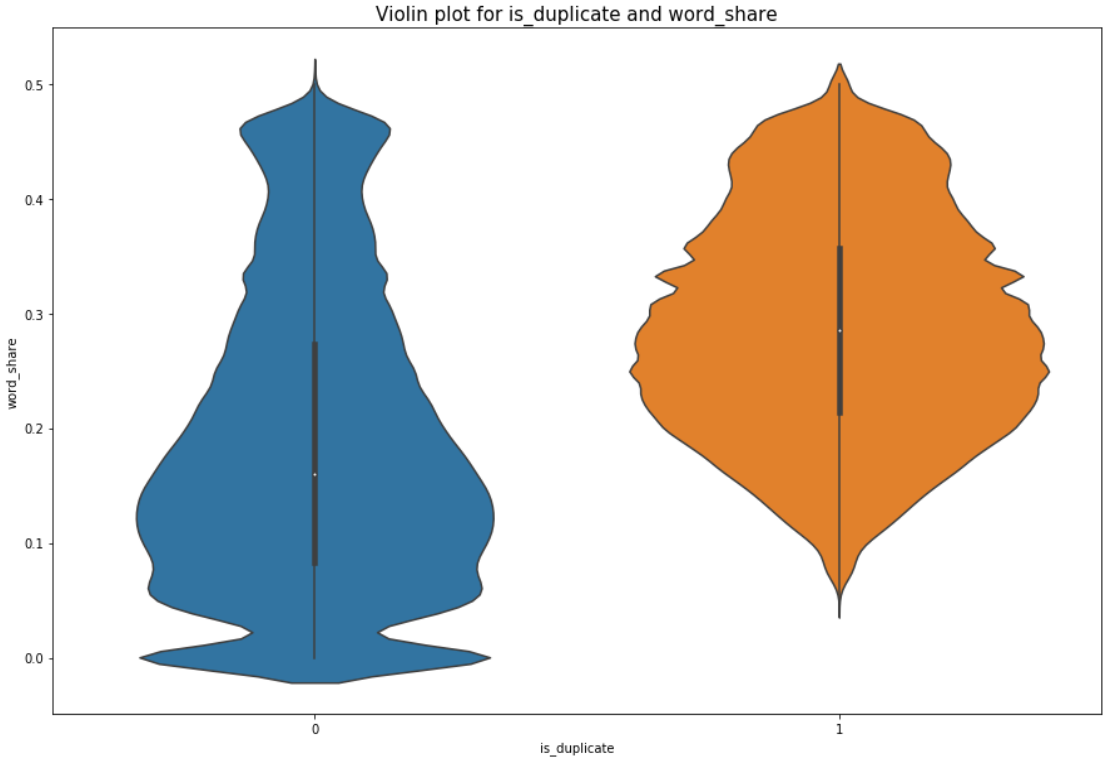
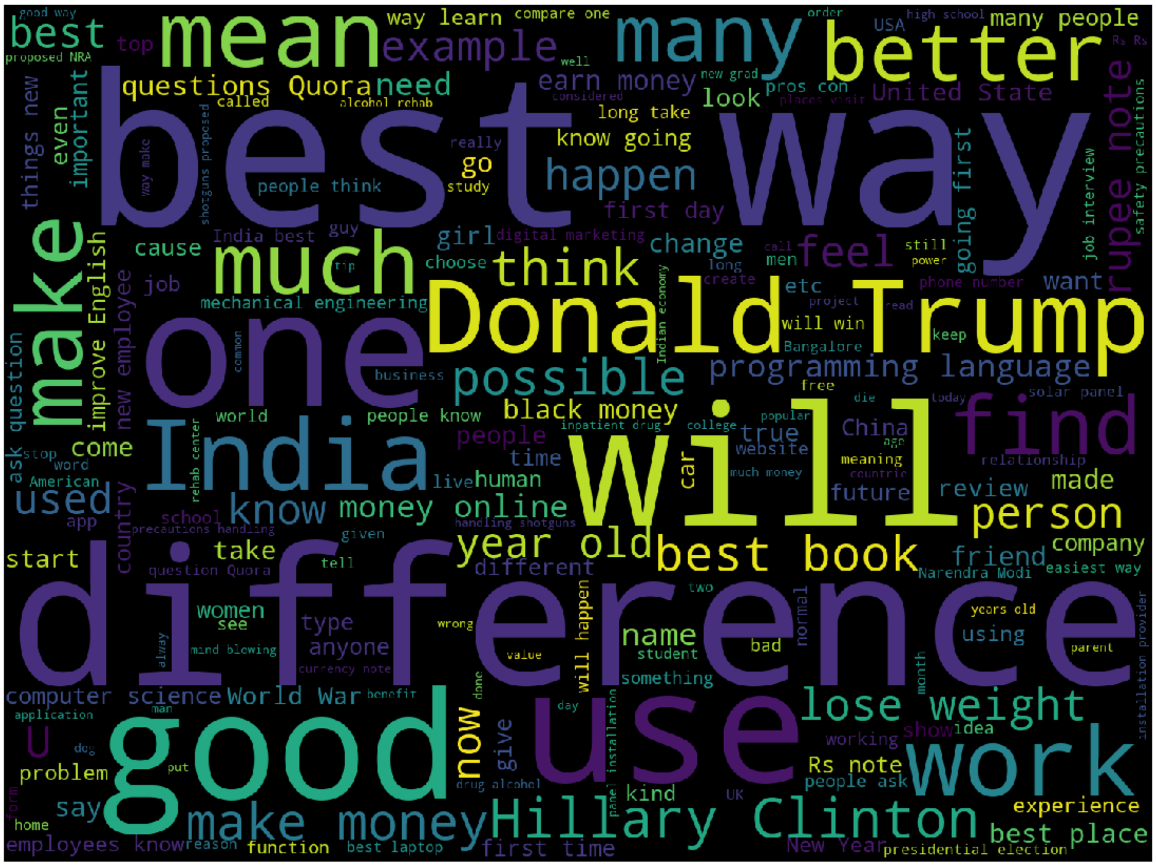


Figure 4: word match share and corresponding violin plot

From the word cloud below, we can see what topics are heated in year 2016. For instance, during the presidential election, donald trump and hillary clinton are heated topics. I also noticed that question involved 'india' takes a large portion of the total questions. After checking its frequency in the basic semantic analysis, I realized the number is obviously too large to be true. I may infer the dataset is constructed with some unique standard, and that further confirms my thought that the distribution of the training data might not follow the true underlying distribution of real Quora's situation.



Questions with question Marks	99.87%
With '.' stops in the middle	6.31%
With capitalized first letter	99.81%
With numbers	11.83%
With 'india' mentioned	4.97%

Figure 5: word cloud and basic semantic analysis

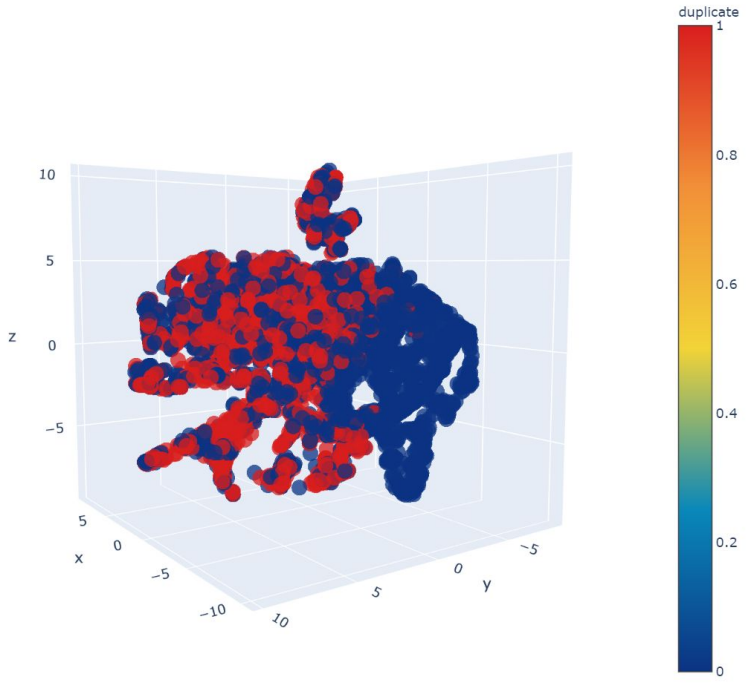


Figure 6: 3D T-SNE Representation

I engineered several features including questions' length, questions' word counts and shared words between the pairs and mapped them into a 3-D T-SNE space. The image agrees with our previous findings that the a big portion of the non-duplicate questions are easy to separate, while the other half is much harder to differ even in a 3D space, which can be beyond the scope of traditional statistical learning. This complicated part requires a more delicate representation, so it can be a good time to introduce neural networks into our study.

Related Work

Some important works for this task are listed below. Even though it's an industry oriented task, the academia shows a strong interest in the problem. The first column shows the model used, the second column shows the word embedding representation and the third column shows their gold metrics - accuracy. Among them, the yellow colored 'Siamese-LSTM' is generally considered as a milestone for this task, because all the other models to some extent refer to the architecture of the siamese neural network, and partly made some trivial adjust to the model.

Model	**Word Embeddings**	**Accuracy**
BiMPM	GloVe (840B tokens, 300D)	**0.88**
LSTM with concatenation	Quora text corpus	0.87

LSTM with distance and angle	Quora text corpus	0.86
Decomposable attention	Quora text corpus	0.86
L.D.C	GloVe (840B tokens, 300D)	0.86
Multi-Perspective-LSTM	GloVe (840B tokens, 300D)	0.83
Siamese-LSTM	GloVe (840B tokens, 300D)	0.83

Figure 7: related models and their performances

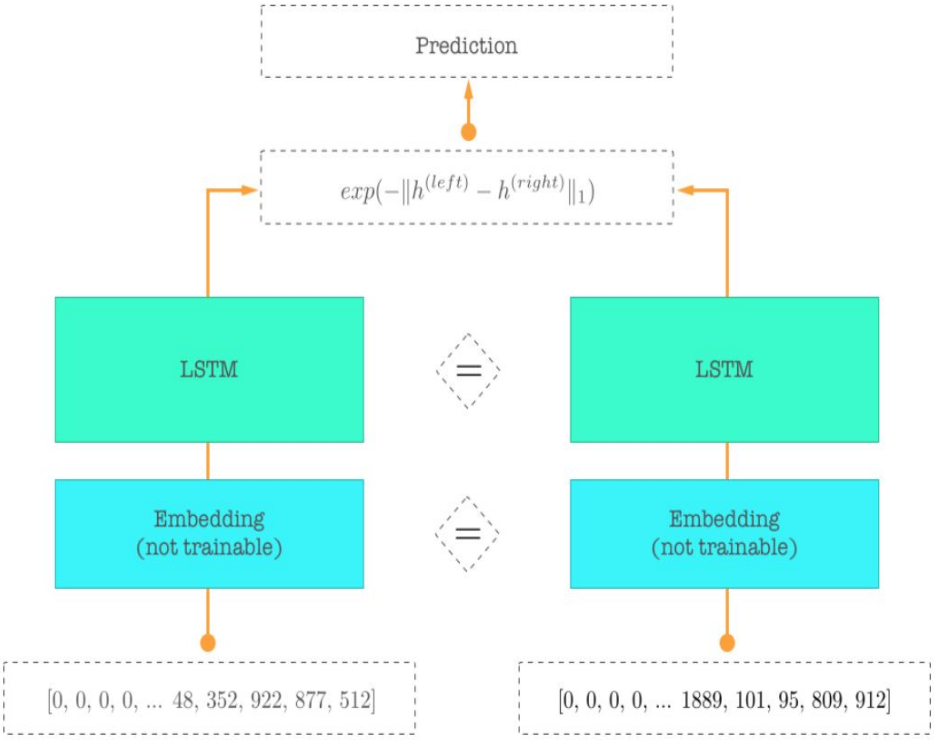


Figure 8: an example of Siamese LSTM

A siamese neural network is a NN that uses the same weights while working in tandem on two different input vectors to compute comparable output vectors. It is easier to train because it shares weights on both sides. As shown in *Figure 5*, the same color blocks shared the same weights and are trained parallelly. It basically consists of two or more identical sub-networks, each taking one of the input representation. The last layer then fed them to a contrastive loss function, which calculates the similarity between the two inputs. There are a lot of practical usage of the siamese neural network such as fingerprint detection, handwriting recognition and

face recognition. It's also commonly used for one-shot learning where people have a limited amount of data, but would like to have some basic knowledge of the data representation.

Baseline

	word_match	tfidf_word_match	q1len	q2len	q1_n_words	q2_n_words
399613	0.500000	0.493545	41	46	7	9
159754	0.800000	0.799257	23	39	4	6
292844	0.444444	0.433485	46	114	9	21
209008	0.363636	0.380526	57	59	11	14
404209	0.727273	0.683345	48	56	10	11

Figure 9: engineered features for XGBoosting

Several features are engineered for the XGBoosting. XGBoosting is known for achieving high scores in various data science competitions. I incorporated the above features - word share, tf-idf word match, the question word length and the question character length. It reached an acceptable score, 0.75 on the validation set, which I would use as the benchmark for this study. According to the Quora's announcement on Kaggle, in the year 2017, they are also using ensemble method - random forests to predict the duplicate questions, but the difference is that engineers in Quora created hundreds of statistical features to be put into the random forest.

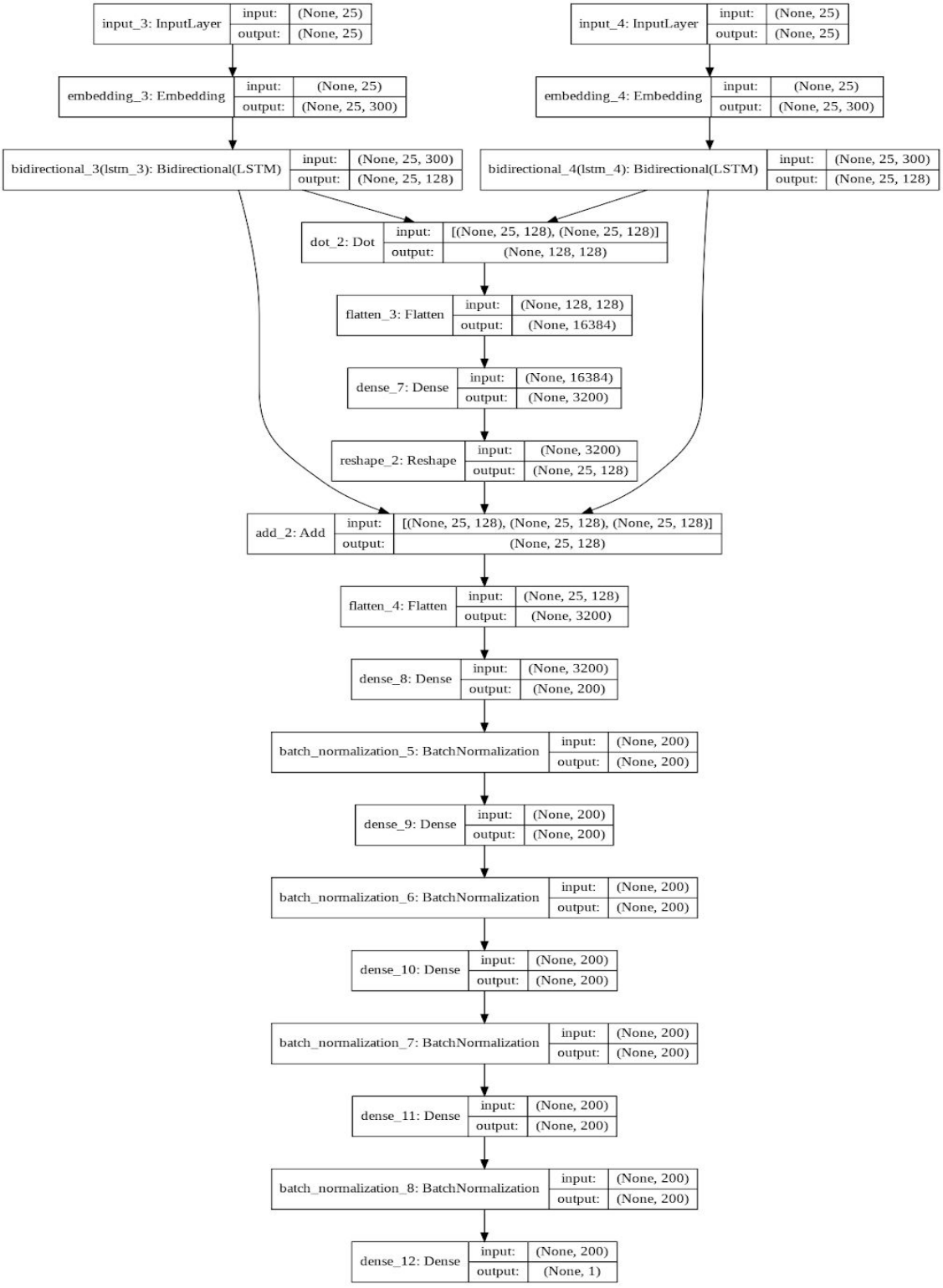
Experiments

1. **Setting**

The data is divided into training and testing with 0.9, 0.1 retrospectively, and then further divided into training and validation 0.9 and 0.1. With this setting, Around 80% of the total data is used for training purposes. After that, experiments for hyperparameter tuning would be conducted on the development data to pick up the final model. Finally, test data will only be used once to evaluate our final model and compare the result to the benchmark.

2. **Modelling**

Figure 10 below displays the full structure of my customized neural network. It contains two raw sequence inputs. The single dimensional vectors are converted into pre-trained GloVe word embeddings (840B 300D). The embedding matrix for each question is then passed to be encoded parallely with bidirectional LSTM. Bidirectional LSTM is a 2-layer LSTM, one of which is responsible for memorizing past information through forward feeding and the other is responsible for recording future information with backward feeding. It is adapted because it's known for behaving well in text classification problems. In the middle part of the sub-branches, an attention layer is added as the dot product of the sequences to emphasize the significant part (the attention) in the sequence formed matrix. After that, the merge layer concatenates the three representations together, reshape them, and pass it to the dense layers afterwards.



\

Figure 10: LJ-LSTM model structure

The model used relu for its activation function to deal with the vanishing gradient problems. Since it's a binary classification problem in nature, it uses sigmoid for last layer's activation function and binary cross entropy for the loss function. The adam is selected as our primary optimizer, as it is recognized as converging faster than other optimizers, but it may lack the ability to reach the optimal when trained with a lot of epoches.

3. Hyperparameter Tuning

Experiment	**Accuracy (Validation)**
Without batch normalization	0.822
With batch normalization	0.827
With dropout (0.5)	0.821
With dense(50)	0.825
With dense(200)	0.827
With dense(300)	0.823
With max_seq_length = 40	0.827

Figure 11: LJ-LSTM model structure

The experiment is run for 25 epoches. As for the primary model, each epoch takes around 200 seconds on Colab's GPU, so one setting would take less than 2 hours to complete. However, changing some hyperparameter would affect the training efficiency greatly. For example, adding the 0.5 dropout would decrease the overall training time to slightly over 1 hour. Changing the max sequence length from 25 to 40 doesn't help improve the accuracy, but would almost double the training time. In addition, we noticed that those settings actually gave a very similar accuracy score on the validation data. One possible reason can be the model structure is in a dominant place for solving this problem. Fine-tuning the problem is definitely useful, but a well-constructed model is apparently playing a more significant role.

Results

As observed from the experiment, we learned the following lessons: firstly, batch normalization increases accuracy. Secondly, dropout isn't helpful in this task, but can improve the efficiency a little bit. Thirdly, using 200 units in the dense layer gives the best result. And fourthly, increasing max_seq_length doesn't improve the accuracy. So our final model comes with batch-normalization, 25 max_seq_length and 200 units in the dense layer. It converges well after 25 epoches' training.

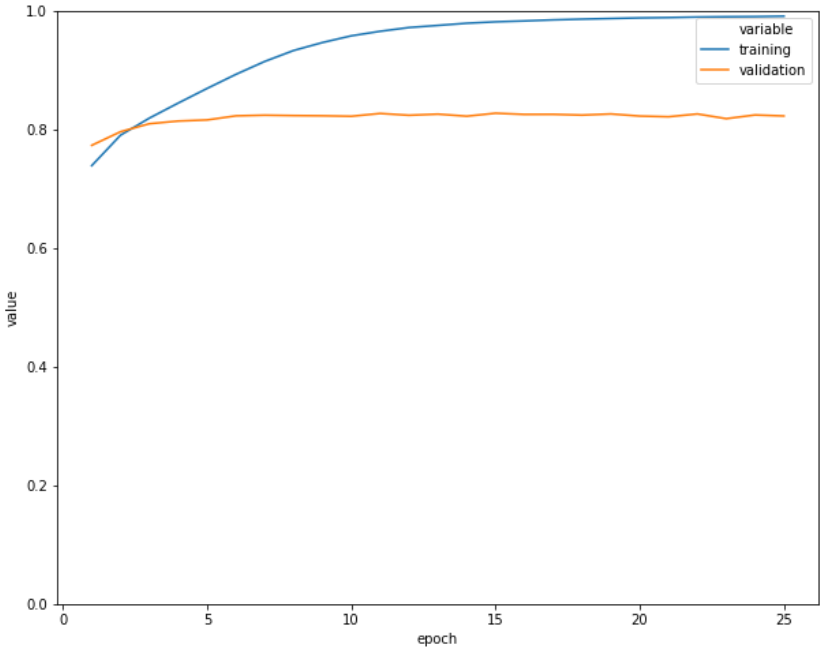


Figure 11: final model performance

It gives a final accuracy of 0.828 on the test data. Unfortunately, it didn't beat the benchmark in academia. However, since I haven't exhaust the preprocessing step, neither have I utilized the randomized search of the hyperparameter space, I do have the confidence this model could beat the 0.83 given more adjust and computation power.

Conclusion

Deep learning methods clearly outperform the XGBoost baselines for the task of duplicate question pair detection. My customized version of siamese network achieved around 8% performance gain over a traditional ensemble method. In short, using this siamese network architecture provides good looking result, and the multilayer had success at concatenating the attention with two bidirectional LSTM.

In terms of possible extension to this study, I suspect using word embedding trained on Quora text data could provide a better behavior. Besides, a deeper cleansing in the preprocessing phase can also be meaningful. So far, I only removed the non-numeric and non-character parts of the text, lowercase characters and tokenized words. Conditions permitting, it would also be worth conducting randomized search over more hyperparameters such as batch size, the repetition rounds of dense layers, the length of sequence represented by LSTM and so on. Furthermore, not much research on detecting duplicate questions in Stack Overflow has been done, and I believe most studies for the Quora problem can be useful for reference.

Reference

1. Lili Jiang, Shuo Chang, and Nikhil Dandekar. "Semantic Question Matching with Deep Learning"
2. Matthew Honnibal. "Deep text-pair classification with Quora's 2017 question dataset"
3. Zhiguo Wang, Wael Hamza and Radu Florian. "Bilateral Multi-Perspective Matching for Natural Language Sentences"