Identify Similar Question Pairs: Quora Question Pairs Challenge

Zhipeng Zhu
University of British Columbia
Zhipeng.zhu@stat.ubc.ca

Abstract

In this article, I present a practical realization of Word2Vec models for sentence similarity by using CNN and magic feature extractions. The data comes from Kaggle challenge "Quora Question Pairs" and contains quite imbalanced classes in both training and test set. I present the techniques and improvements we could use to improve the training performance in three aspects: preprocessing, model structure, and feature extractions.

1 Background

Every year there are millions of questions posted on the Quora website. Quora provides a perfect platform to ask questions and link people with similar interests. As the number of questions posted increases dramatically, more and more questions with similar content exist on the platform at the same time. Users who are seeking for answers may spend more time finding the best answer under the most appropriate question. The writers will need to answer several versions of answers for questions with similar content. Even for those regular readers on the platform, too many duplicated questions indeed decrease the reading efficiency. (Kaggle, 2021)

Hence, Quora posted a challenge of identifying questions that have the same intent on Kaggle. This challenge provides a rich dataset of the potential duplicated questions in Quora and asks for advanced techniques to classify whether the question pairs share the same intent. Questions are well organized in form of pairs and the labels are provided.

2 Introduction

There are dozens of methods and toolkits for calculating the sentence similarities. Some of them are based on the syntactic structures of the sentences. This type of calculation includes edit distance, Levenshtein distance and Jaccard index. The other methods, such as Cosine distance based on Word2Vec, relies more on pretrained models and would consider more semantic meanings of the words.

The idea of Word2Vec is that the words occurring in the same context will probably have similar meanings. It takes the input texts and produces a vector space with hundreds of dimensions. The Word2Vec assigns each word with a single vector which has been specified in the vector space. Since a sentence is composed by several words, the sentence can be vectorized by adding the single vectors in it. The cosine values can be calculated based on the angle of the vectors of the sentence pair. The higher the cosine values are, the more similar the sentences are.

However, for many sentences in the reality, the cosine values of the vectorized pairs cannot precisely indicate the true similarity between pairs. Therefore, the Word2Vec model usually acts as a method of feature extraction in projects focusing on sentence similarities. Practically, the Word2Vec helps construct the embeddings for the rest of the model.

There are two main methods to utilize the Word2Vec embeddings, the one is Long Short-Term Memory, LSTM approach and the other is to use convolutional neural network. In this article, both approaches will be realized and compared as baseline models, while further feature extractions will be applied to improve the performance and accuracy.

The main objectives of the project are summarized as following:

- Present techniques and concerns of preprocessing input sentences.
- Construct baseline models with LSTM and CNN structures.
- Explore different feature extractions and compare their performance benefits.

The evaluations of the models will be based on the training accuracy, test accuracy and running time. The LSTM and CNN baseline model will be two baseline

models and be compared first. Then with similar added features, I will bench the improvements on both accuracy and running to select the best magic features.

3 Data Overview and Preprocessing

The data comes from the Kaggle competition "Quora Question Pairs". It includes the presplit train set and test set. For competition requirement, the test set is 5 times larger than the train set. Both datasets are not clean and need preprocessing for usage in models. A brief exploratory data analysis is necessary before the modeling.

3.1 EDA

Here the table.1 is a summary of the text data:

Columns	Data Type	Notes
id	Integer	The sequence number of question pairs.
qid1 qid2	Integer	The specific sequence number for each question.
question1 question2	String	The actual questions in pairs.
is_duplicate	Binary	Indicate true or false of whether pairs are duplicate.

Table 1, Data Summary

Data Set	Size
Train	404290
Test	2345796

Table 2, Size of Dataset

While the train set and the test set differ a lot in sizes, their distributions of word counts in pairs are quite similar. Here is the normalized histogram for word count in questions:

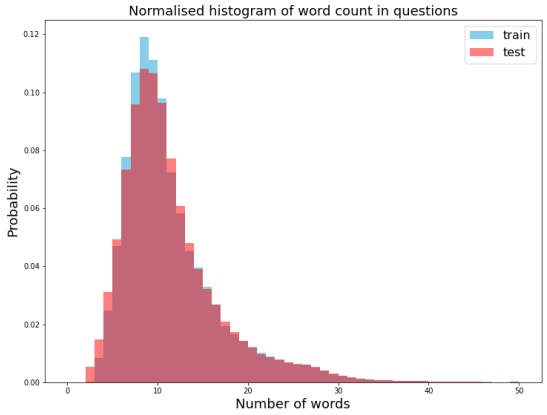

Figure 1, Normalized Histogram of Word Count

The average word count in questions is around 11 and most of the questions has the words in a range of 5 to 20. Hence, even Word2Vec may perform better in short texts, the length of the sentence in our data may still in the capacity of Word2Vec. (Kusner, 2015)

3.2 Preprocessing

As most of the text data require preprocessing, the question pairs from Quora need quite a few steps to clean up.

Lowercasing: The questions are kept in the original state as we read it. Lowercasing is one of the simplest techniques of text preprocessing which helps deal with sparsity issues in the data. It eliminates the variations of words caused by mixed-case occurrences. For example, without lowercasing, "What" and "what" will be recognized as two words. In this sense, lowercasing reduces the vocabulary size of our input text. Despite the benefits in sparsity reduction, the lowercasing may also cause more ambiguity. A global lowercasing will have all the words which belong to specific entities, such as brands, architectures and geographic locations, transformed into the normal words.

Stemming: Stemming is a process of removing inflectional forms of words based on elementary rules. It relates and maps the words to the same stems and reduce all these words to the elementary stem. Since this technique will reduce several words into one stem, it greatly reduces the vocabulary size of text. However, most of the stemming algorithm may also generate non-meaningful stems. There is high probability that it will generate stems such as "decreas" which loses its meaning and is not found in our word vector files.

Stopwords: There are many frequently used words in many sentences, including "a", "is" and "the". In many cases, without these commonly used words, the left part of the sentence will still present its own concentration in meanings. This technique will leave the more important words and prevents the other common stopwords being analyzed. For example, the sentence "It is a preprocessing technique" will become "preprocessing technique" after removing the stopwords "it is a".

Hardcoded Rules: Besides the above global processing techniques, hardcoded rules will help in dealing with specific issues in the dataset. There are three commonly used hardcoded rules for texts, including contractions, punctuations and abbreviations. These rules reshape the sentences and reduce the vocabulary size while limit the potential effects in increasing ambiguity.

4 Models

As shown in Figure 2, my system performs on LSTM or CNN models based on the embedding layers from Word2Vec adding the further extracted features.

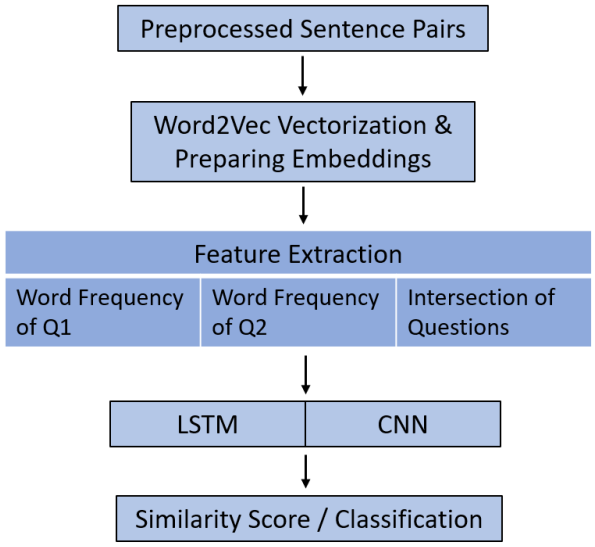

Figure 2, System Structure

There will be two main layers as the input the LSTM and CNN models. One is the embedding matrix produced based on Word2Vec and one is the added features. The LSTM and CNN layers will be set as similar as possible in order to compare their performance.

4.1 Word2Vec and Feature Extraction

First of all, the all the sentence in question 1 and question 2 will be converted into sequences. We define the maximum count of the word in sentences and these sentences which have fewer words will be padded with zeros to match the maximum length. The question pairs will be split into sequences for both questions in pairs.

Then the prepared "GoogleNew_Vectors_Negative 300" will be used as the embedding file for Word2Vec. This file contains the pretrained Google News corpus with 3 billion running word vector model. Based on the pretrained vector model, we than create an embedding matrix for the following models.

With the input sequences of questions, we have several methods to extract the features. There are some basic features such as the word counts, frequency of each question and the word count of intersections. Some advanced features may also be utilized, including TFIDF over character and word ngrams, fuzzy string matching and word mover distances. In this system, I will only use the frequency of questions and the word count of the intersections between pairs as the extracted features.

4.2 LSTM

The LSTM, Long Short-Term Memory, is an artificial recurrent neural network architecture. For the LSTM module, we will have two parallel inputs for two sequences from question 1 and question 2. The two prepared sequences of questions will be put into two embedding layers according to the size of the embedding matrix produced before. Then these two embedding layers will be the input of two LSTM layers accordingly. In this system, I use 200 as the number of units for the LSTM layer.

As the two parallel embedding layers and the LSTM layers are created, we will have the two LSTM layers merged and assign the dropout rate, dense sizes and activation function to them. The LSTM model can be realized by two different ways, one is using the cuDNN acceleration and the other one is using the generic one from GPU. To fulfill the requirement of cuDNN, our dropout rate will be set as 0 and the "tanh" function will be the only choice for activation. Instead, we may prefer use 0.2 as the dropout rate and the "ReLu" function for the activation. Both GPU modules can deliver a descent running process for LSTM models, while the running time and accuracy may vary.

The extracted features will also be transmitted into the model as a feature layer. In the LSTM baseline model, this feature layer will be deleted, and the input of the model will be only the two sequence of question 1 and question 2.

4.3 CNN

The CNN, Convolutional Neural Network, is commonly used in visual imagery analysis. (Valueva, 2020) It is also a popular model architecture for natural language processing tasks.

Similar to the LSTM module, the CNN will have two parallel embeddings from the two sequences of question 1 and question 2. However, it is important to define the needed layers and the corresponding configurations for 1D CNN module. In our model, we will use 6 1D CNN layers (kernel size from 1 to 6) and 128 as the number of filters for each layer. The dropout rate will be stay at 0.2. The activation function will use "ReLu" as well.

The extracted feature will participate in the module as a feature layer with 3 dimensions. The baseline model of CNN will discard the extracted features and leave the input with only two sentence sequences.

4.4 Evaluation

The system will simply use the running time, train/validation loss, and the train/validation accuracy of

baseline and feature-added model as the metric for evaluation.

5 Results

Both baseline models and feature-added models are running with the same inputs (the sentence sequences from question 1 and question 2). While the LSTM models will run on two GPU modules, one with cuDNN and one with the generic module. The results are summarized in Table 3 and Table 4.

Baseline Model	LSTM (cuDNN/generic)	CNN
Train Accuracy	0.883 / 0.852	0.924
Train Loss	0.177 / 0.223	0.312
Val Accuracy	0.825 / 0.851	0.860
Val Loss	0.312 / 0.274	0.327
Running Time	76s / 860s	154s

Table 3, Training Results of Baseline Models

Feature-Added	LSTM (cuDNN/generic)	CNN
Train Accuracy	0.899 / 0.857	0.923
Train Loss	0.156 / 0.205	0.123
Val Accuracy	0.822 / 0.852	0.853
Val Loss	0.304 / 0.271	0.292
Running Time	88s / 920s	160s

Table 4, Training Results of Feature-Added Models

All the presented results are from the models with the input that only have hardcoded rules.

5.1 Preprocessing

With the stemming function from NLTK model, both the LSTM and CNN models will run about 5% less time than the presented models. Under stemming inputs, both the training accuracy and the validation accuracy will decrease by 3-4%. Meanwhile, with the stopwords removal, the model training has an additional 2-3% acceleration. Compared to the stemming, stopwords removal will only cause less than 2% decrease in accuracies.

The side effects from stemming and stopwords removal do not limited to the running times and accuracies of models. They also consume quite noticeable longer execute time to preprocess the input questions. Though the running time of model itself will be shorten when we apply these preprocessing techniques to the sequences, with the longer execute time, these two techniques seem to be quite uneconomic and impractical.

5.2 Model Structure

Despite the effects from the preprocessing techniques, we may still be able to compare the models based on the reported results.

First of all, the LSTM model performs quite different with the different configurations due to the GPU modules. With cuDNN, LSTM gains a super-fast running speed while the training accuracy becomes way higher than its validation accuracy. However, with the generic GPU module, LSTM provides a quite close and relatively high training accuracy and validation accuracy. In this sense, it seems that with "tanh" activation function and 0 dropout rate, the LSTM model sacrifices its validation for the running speed. Meanwhile, the higher training accuracy to the validation accuracy implies that the cuDNN LSTM is overfitted. The generic LSTM model seems to be a more valid training model than the cuDNN one, and it even achieves a comparable accuracy to CNN model. The CNN models have a fast training speed as well, and it keeps a high training accuracy and a validation accuracy. Though it is still overfitted, CNN could be a better choice compared to LSTM, considered its running time and accuracy.

5.3 Feature Extraction

The feature layers in this system have very minor effect to the training, compared to the preprocessing techniques and model structures. With the feature added, the models will require more running time generally. The overall accuracies will even decrease when we add the extracted features. It seems that the extracted features do have benefits to the training performance. However, we notice that with the added features, both the training loss and the validation loss of CNN models have significant decrease. In this sense, the added features make the models more valid.

6 Conclusion

Based on the results reported from the model training, we may figure out some clues of which preprocessing techniques are more suitable for the sentence similarity task. The hardcoded rules on word contractions, punctuations and abbreviations works well for training while we will have to deal with the trade-off from both the stemming and stopwords removal.

ctions should go as a last section immediately before the references. Do not number the acknowledgement section.

Provided the performance of LSTM and CNN based models, all three training methods may have its own advantages. The LSTM with cuDNN is probably the best model to choose when we need its real-time response and can compromise on its relatively low accuracy. The LSTM with generic GPU module may be the most valid model though it needs much more time consumption than both cuDNN LSTM and CNN. The CNN model has high potential to become the sweet point between cuDNN LSTM and generic LSTM.

Current feature extractions, such as frequency of questions and word count of intersection between pairs, has shown its improvements on model validation. Considered there are more extraction methods, including TFIDF, Fuzzy matching and mover distance, we can expect more performance benefits from added features.

7 Future Works

7.1 New Model Structures

The LSTM and CNN models have achieve acceptable results while we can still have more improvements if we take new structures in. For example, we may use the CNN as one layer before the LSTM layer. In this sense we will create a CNN-LSTM model which may achieve even better training performance.

Besides the CNN-LSTM model, we may also try Transformer or BERT to solve the sentence similarity task.

7.2 New Feature Extractions

In this paper we use three basic features extracted from the questions. We anticipate having more performance benefits from advanced features. Here is a list of potential useful feature extractions.

- TFIDF of character and word
- Fuzzy matching
- Number of characters
- Word mover distance
- Skew/kurtosis of the sentence vectors

References

Kusner, M., Sun, Y., Kolkin, N. & Weinberger, K.. (2015). From Word Embeddings To Document Distances. *Proceedings of the 32nd International Conference on Machine Learning*, in *Proceedings of Machine Learning Research* 37:957-966 Available from https://proceedings.mlr.press/v37/kusnerb15.html.

Quora question pairs. Kaggle. (n.d.). Retrieved November 17, 2021, from https://www.kaggle.com/c/quora-question-pairs

Valueva, M.V.; Nagornov, N.N.; Lyakhov, P.A.; Valuev, G.V.; Chervyakov, N.I. (2020). "Application of the residue number system to reduce hardware costs of the convolutional neural network implementation". Mathematics and Computers in Simulation. Elsevier BV.177:232–243. doi:10.1016/j.matcom.2020.04.031 ISSN 0378-4754. “Convolutional neural networks are a promising tool for solving the problem of pattern recognition.”

Appendix

Data source:

Quora Question Pairs | Kaggle

Code in GitHub:

zhuzp98/Sentence-Similarity-Experiment (github.com)