QUORA QUESTION PAIRS

A PREPRINT

Anil Ozdemir
Department of Computer Science
Sabancı University
Istanbul, TR
19453
aozdemir@sabanciuniv.edu

June 20, 2020

1 Introduction

Quora is a platform for asking questions. Where you can find qualified people, who have unique ideas and quality answers. Every month over 100 million people visit Quora so this is not surprising that there are multiple questions with similar meanings and same answer. Detecting these duplicate questions can greatly be helpful for seekers to find their answers without spending too much time [1]. Looking through the literature there are multiple studies that focused on paraphrase detection in Natural Language Processing (NLP). They can be classified into two groups. One group used machine learning techniques and methods for this aim and other group used deep learning to detect the paraphrases. The most recent studies used deep learning. Recently, the methodology of the NLP completely shifted from the classical approaches to the neural network based models .[2]

In paraphrase detection, some studies did not use a lot of pre-processing for not losing any information [3]. But there are some routine pre-processing methods that are done in almost all of the text analysis studies. For example, the words or sentences are tokenized and vectorized and characters are changed to lowercase. For using machine learning methods the most challenging phase is the feature extraction. Studies have used different kind of features for implementing their models. In Chitra's paper, they had lexical features and semantic features. For extracting lexical features they used skip-grams and for extracting semantic features they used word net corpora [3]. In another study, they used twitter data for paraphrase identification, the features that they used were size of union, size of intersection and text size [4].One of the most common approaches for feature extraction is using TF-IDF method. In information retrieval, TF-IDF, short for term frequency inverse document frequency, is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus [5]. In Lei Guo et al, after pre-processing and vectorizing all words using Google's Word2Vec model [6] they multiplied each word vector with its TF-IDf weight to represent each question with corresponding vector. They also used number of common words in two questions and difference in length of two questions as their features [7].

As discussed before, studies are classified in two groups as machine learning focused and deep learning focused. The most common machine learning classification methods that are used are SVM, Logistic regression and Naïve Bayes. In Chitra's paper [3], they used SVM classifier with linear, polynomial, radial and sigmoid kernels. The results showed that the radial basis kernel has worked well for paraphrase identification with accuracy of 70%. Eyecioglu et al used SVM as well, in their study linear kernel performed better [4]. In Gabbard et al's study, they implemented three different methods including logistic regression, Naive Bayes and recurrent neural networks. Their best model was recurrent neural network which has been very useful in text classification problems [8]. Yin et al, presented a deep learning architecture using convolutional neural network. Their results showed that using convolutional neural network improved their model's performance [9]. In this project, we used Quora Question Pairs data set. The training data set includes 404,290 lines of question pairs and an ID for each line. Questions also have their specific IDs. There is a target variable which have binary values of 0 or 1. 0 indicates that questions are not duplicate and 1 means that questions have same meanings. The test data set includes 2,300,000 question pairs without labels. We only used training data set in our analysis and we split it into training, validation and test data set. We took question pairs as our variables and "is

duplicate" column as our target. We applied a few pre-processing methods, such as lower casing the text, removing stop words, punctuation and symbols. We generated the features for implementing logistic, Random Forest and XGBoost machine learning algorithms. The literature showed that deep learning performed better on paraphrase identification. To examine this fact we also implemented convolutional neural networks on our data set.

2 Task Description

In this study, we worked on identification of question pairs that have the same intent. This is also a high level competition known as Quora Question Pairs organized by Kaggle [1].

Our project is a binary text classification problem that gets the question pairs as input and decides whether that questions express same meaning or not.The Following two labels assigned to particular question pairs as a result of binary classification.

- Similar :
  Question pairs that express same meaning.
- Not Similar :
  Question pairs that not express same meaning. .

3 Dataset: Quora Question Pairs

In this work, we used Quora Question Pairs Data-set which provided by the Kaggle and sponsored by the Quora. Data consist of 404,290 labeled training sample with 37% of similar question pairs and 63% of not similar question pairs. We split it into training, validation and test data set (12.5% as test data). Each sample of the training has the following features:

- id: The id of a training set question pair
- qid1,qid2: unique ids of each question
- question1, question2: The full text of questions
- isduplicate : The target feature, labeled by 1 if question1 and question2 have originally the same meaning, and 0 otherwise [1].

4 Methods

In this part, we present the different architectures, algorithms and pre-processing approaches we tried, together with a comprehensive description of the designs of classifiers.We followed the below scheme during project:

1. Pre-Processing Feature Extraction
2. Machine Learning Baseline Models:
   - 3.1 Logistic Regression Classifier
   - 3.2 Random Forest Classifier
   - 3.3 Xgboost Classiferr
   - 3.4 Logistic Regression Classifier
   - 3.5 Random Forest Classifier
   - 3.6 Xgboost Classiferr
3. Deep Learning Models.
   - 5.2 Long-Short term Memory (LSTM).
   - 5.1 Convolutional Neural Network (CNN).
   - 5.1 Convolutional Long-Short term Memory Network (CNN-LSTM).

4.1 Pre-Processing

Recently, previous studies have stated that different pre-processing methods have a significant effect on the performance of classifiers. These methods can either improve the classifier's performance with reducing the dimensionality of data or cause significant loss of variance [10]. The pre-processing procedures that we applied to the data are as follows:

- Tokenizing the questions
- Converting all text to lowercase
- Removing all symbols
- Removing all punctuation
- Removing all stop-words

4.2 Feature Extraction

In first part of the project the extracted features from the data were as follows:

- Word Mover's distance among questions:
  We vectorized the questions using Google's pre-trained Word2Vec model from Gensim library in order to make distance calculations possible. As a next step, we calculated the word distances by using Word Mover's Distance which is a measure that shows the dissimilarity between two documents as the minimum travel distance from embedded words of one document to the embedded words of the other document [11] and used this distance metric as one of the features in our models.
- Question frequencies
- Number of common words between question pairs
- Term Frequency–Inverse Document Frequency (Tf-Idf):
  Firstly, We built the Tf-Idf matrix with specific cut-off value (we only take 10000 highest value) for both question 1 and question 2 from vocabulary which obtained only from train data-set. Afterwards, we apply concatenation to both two matrices and insert that sparse matrix as a feature in the machine learning models with other features.

We updated the features and used new features in the final phase of the project. We added new features to our feature set and re run the models. Using these new features improved the computation time and performance of the models. Final list of features are as following.

- Number of shared unigram word between questions
- Shared Tf-Idf score
- Word Mover's distance among questions
- Jaccard similarity coefficient:
  The Jaccard Index, also known as the Jaccard similarity coefficient, is a statistic used in understanding the similarities between sample sets. The measurement emphasizes similarity between finite sample sets, and is formally defined as the size of the intersection divided by the size of the union of the sample sets.
- Cosine distance
- Frequency of questions
- Length of questions
- Length difference between questions

4.3 Machine Learning Baseline Models:

We applied 3 ML methods which are stated on the following sub-sections:

4.3.1 Logistic Regression Classifer

Logistic regression model that we used in our task is exploited by Scikit-learn library [12]. In trianing part, we fine-tuned the model with Randomized Search for finding optimal hyper-parameters. Moreover, we applied cross validation to avoid overfitting. We only used 20,000 question pairs for parameter tuning, because of computational time limits.

4.3.2 Random Forest Classifier

Random Forest model that we used in our task is exploited by Scikit-learn library [12]. In training process,we optimized the model with Randomized Search for finding optimal hyper-parameters. For a Random Forest, there are more hyper-parameters that can be adjusted in comparison to the other models. We particularly focused on number of estimator and number of depth since they are the most significant hyper-parameters in terms of both time efficiency and overfitting. Since increasing number of trees causes overfitting, we also performed hands-on tuning on the random forest model. In addition, we applied cross validation.

4.3.3 XGBoost Classifier

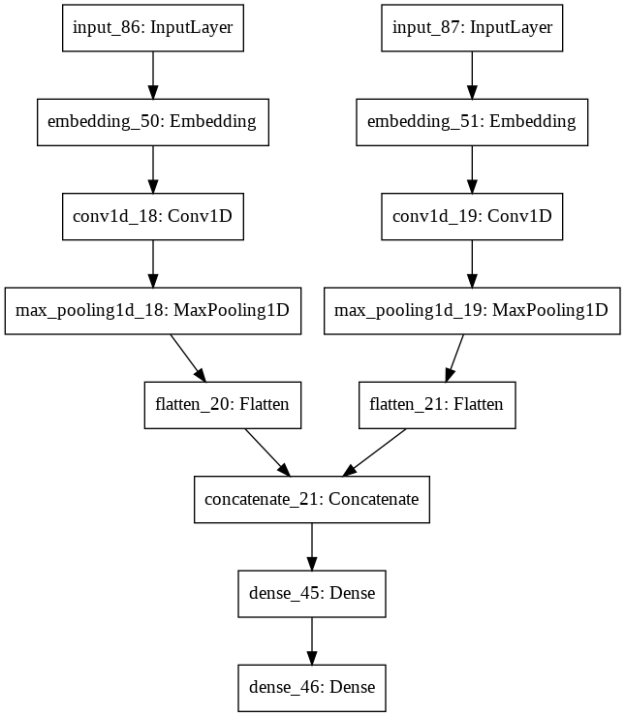
XGBoost is a tremendous tree boosting algorithm, which was presented by Tianqi Chen in 2016. It stands for extreme gradient boosting. There are two reasons why we chose to use this algorithm, which are execution speed and model performance. XGBoost is one of the fastest algorithm when compared to other implementations of gradient boosting [13].The implementation is supported by xgboost library. We also tune the model hyper-parameters with Randomized Search on multiple validation data set from the cross validation.

4.4 Deep Learning Based Models

We implement a Convolutional Neural Network model and Bidirectional Long Short Term Memory Network model as deep learning models.

4.4.1 Convolutional Neural Network with Pre-trained Word Embedding

Figure 1: CNN Model



As shown in Figure 1, the CNN model in this project is a multi-channel (2 channel, one per question) sequential model that starts with the embedding layer which takes pre-trained word embeddings from word2vec [6] as non-static embedding.Then, concatenates and maps them to a dense vector by using Gensim's functionality. After that, the output from the embedding layer is going to be an input for the convolutional layer with 128 filters and 3 kernel size. Then 1-dimensional global max-pooling layer which is used to connect convolutional structure with traditional fully connected layers.

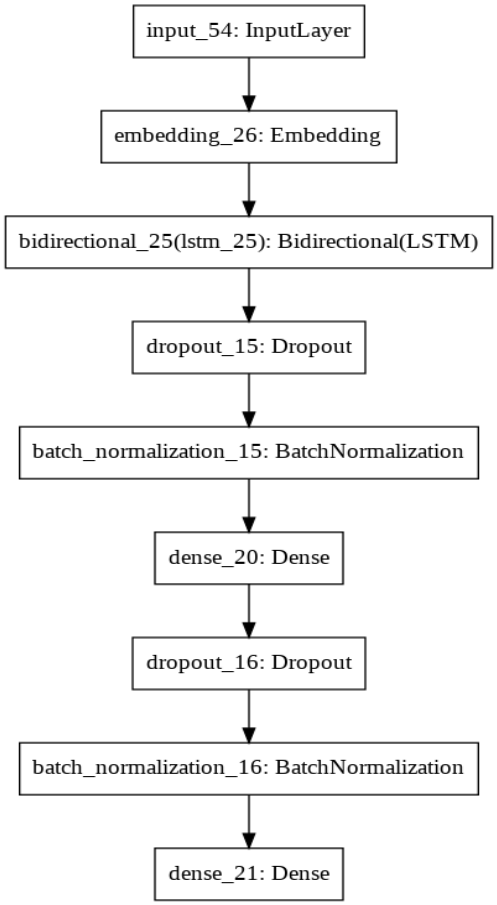
Model	Test Precision	Test Recall	F1	Test Macro F1	Test Accuracy	Train Accuracy
Logistic Regression	0.75	0.52	0.61	0.72	0.76	0.83
Random Forest	0.89	0.31	0.46	0.64	0.73	0.84
XGBoost	0.77	0.37	0.50	0.65	0.72	0.85
CNN	0.6	0.54	0.56	0.55	0.69	0.99

Table 1: Evaluation of Models before feature extraction

4.4.2 biDirectional Long-Short Term Memory with Pre trained word embedding

LSTM is the sub-type of the Recurrent Neural Networks, which is one of the most popular models used in Natural Language Processing due to its recurrent design, is very suitable for texts. RNN's are able to use distributed representations of words by transforming the text tokens into vectors, which is going to be the basis for the representation matrices. BiLSTM is the extended version of the standard LSTM model and it provides accessing past and future information by propagating backward as well as forward [14]. BiLSTM model used in this experiment is depicted in figure [2] and implemented via Keras. It uses pre-trained word embedding in its multichannel embedding layer ad than each of word embedding vectors is fed to the Bilstm.The Output from bilstm is followed by a drop-out layer and two block of batch-normalization layer.Later,output is connected to the fully connected layer(dense layer).Afterwards, extracted logits from the dense layer fed sigmoid activation function.

Figure 2: biLSTM Model



Model	Test Precision	Test Recall	F1	Test Macro F1	Test Accuracy	Train Accuracy
Logistic Regression	0.64	0.32	0.43	0.60	0.68	0.74
Random Forest	0.76	0.34	0.47	0.64	0.71	0.80
XGBoost	0.65	0.41	0.51	0.65	0.70	0.89
CNN	0.69	0.70	0.69	0.76	0.77	0.96
biLSTM	0.69	0.57	0.62	0.72	0.75	0.75

Table 2: Evaluation of ML Models After Feature Extraction plus CNN , biLSTM Fine tuned

5 Results Discussion

In our work, the objective was the identify the two questions such that they have common sense or not. We approach this problem as a binary text classification task that assigns the 1 to the given two questions if they are similar and assigns the 0 if they are not similar. In the initial phase of the project, our first hypothesis was the Tf-Idfscores from both questions are sufficient for estimating such a similarity. However, this preference resulted in scores that were not acceptable in the evaluation metric but also a large computation time. Random hyper-parameter search and training with such a sparse feature matrix took approximately one day per model, especially in tree-based machine learning models. One can see the most successful model was the base logistic regression model with 0.72 test macro f1 score by looking table 1. Besides, we used only word2vec embeddings for the Convolutional Neural Network model in the beginning, but we did not perform any fine-tuning so that the model didn't work well at the beginning.

In the second phase of the project, we used a shared Tf-Idf score instead of entire sparse Tf-Idf scores with adding more additional features such as, Jaccard similarity coefficient, Cosine similarity, number of common unigrams and 10 more features (see 4.2). This approach resulted in a 1% drop in results but gave us about 20 times the speed. On the other hand, deep learning models we used , such as CNN and biLSTM achieved a high rate of success by taking two inputs over different channels using word2vec embedding. We achieved the best accuracy of 77% and a macro f1 score of 76 % with Convolutional Neural Network. With the high rate of the macro-f1 score, we overcome the imbalance problem of the data because we have achieved a close precision score on both labels.

To sum up, the machine learning methods perform a success rate of around 70 percent in this problem and we do not find this rate sufficient, but it is theoretically possible to gather these models under a single ensemble model and to find more successful results. On the other hand, deep learning methods perform more successful and robust results.

References

[1] Kaggle. Quora question pairs, 2017-11.

[2] Mitchell Marcus. New trends in natural language processing: statistical natural language processing. *Proceedings of the National Academy of Sciences*, 92(22):10052–10059, 1995.

[3] Anupriya Rajkumar and Chitra Rajan. Paraphrase recognition using neural network classification. 2010.

[4] Asli Eyecioglu and Bill Keller. Twitter paraphrase identification with simple overlap features and svms. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 64–69, 2015.

[5] Anand Rajaraman and Jeffrey David Ullman. *Mining of massive datasets*. Cambridge University Press, 2011.

[6] Yoav Goldberg and Omer Levy. word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method, 2014. cite arxiv:1402.3722.

[7] Lei Guo, Chong Li, and Haiming Tian. Duplicate quora questions detection.

[8] Akshat Doshi and Jay Patel. Quora insincere question classification.

[9] Wenpeng Yin and Hinrich Schütze. Convolutional neural network for paraphrase identification. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 901–911, 2015.

[10] Emma Haddi, Xiaohui Liu, and Yong Shi. The role of text pre-processing in sentiment analysis. *Procedia Computer Science*, 17:26–32, 2013.

[11] Matt Kusner, Yu Sun, Nicholas Kolkin, and Kilian Weinberger. From word embeddings to document distances. In *International conference on machine learning*, pages 957–966, 2015.

[12] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in python. *J. Mach. Learn. Res.*, 12:2825–2830, November 2011.

[13] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794. ACM, 2016.

[14] Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997.