An Overview for Quora Question Pairs Competition

1st Zhongbo Zhu 2nd Bo Pan 3rd Shuhong Xiao 4th Xinyu Lian 5th Yicheng Lu

***Abstract*—Quora Question Pairs, consider as sub-question of natural language inference (NLI), is to present whether two question could be mapped equal or not. In NLI field, recently works have mostly focus on neural networks based approaches, which have shown to be very effective based on large notable dataset. In this paper, we tried a series of approaches, ranging from deep neural networks (Bidirectional Long Short Term Memory, Convolution neural network and Multilayer Perceptron), traditional model (XGboost)to integrated models. Besides, we did a thorough exploration of features(embedding features, traditional NLP features, stuctural features) and find that feature engineering is the most critical part for this problem. Finally, we combine all the work we did, concatenated all the features we have, and stack all the models together to introduce as much diversity as possible. The result showed that with this approach we can reach a 93.3% final accuracy with 0.15 log loss on the test set (which ranked 279/3296).**

I. Introduction

Reasoning and inference, as an important sub-question in language processing task, is considered a challenge but basic problem. The previous works has included extensive researches mainly in three approaches: A set of artificially written logic and common sense rules used to obtain the mapping of natural language to logical form in the data from Gordon(2016),Traditional artificial intelligence methods based on statistical methods combining word matching and other lexical features with manually written deterministic rules or traditional decision tree and neural network based inference models with large annotated data [12]).

Specifically, our task, known as "Quora Question Pairs", is to present whether two given question could be mapped equal or not. For this task, we consider it as a sub-question of natural language inference (NLI) which concerned with determining whether a natural language hypothesis h can be inferred from a premise p. An example by Quora is shown following:

P: Why does Batman get kill in Batman v Superman?

H: In Batman v Superman, why reduce Lex Luthor pit Superman against Batman?

The most recent years have seen advances in modeling natural language inference. Alex [13] improved the concept of Long Short Term Memory and demonstrating the advantages of advanced recurrent networks over other sequential algorithm. A generalization of LSTMs to tree-structured network topologies was introduced by Kai Sheng Ta (2015) [14] that combine words to phrases based on syntactic properties. Tree-based convolution neural network (TBCNN) also be applied to captures sentence level semantics, heuristic matching layers like concatenation, element-wise product/difference combine the information in individual sentences and shown better performance than other encoding approach (Lili Mou 2016) [15].

While some pervious top model uses rather complex network, in this paper, we demonstrate the Multilayer Perceptron based network, 1 D convolution network and enhanced sequential inference models based on chain model (Qian Chen 2017) [7]. We have all our models achieve an accuracy of about 83%, suggesting that more complex architecture does not notably improve the accuracy of predictions in our Quora task which is a simplified natural language inference (NLI) problem. We also introduce a Committee mechanism and a further improvement in accuracy has been seen compare with these single models, suggest that participation of low-correlation features are more important than model complexity. The traditional feature engineering also highly improves our models performance. We achieved extra 91 features, increasing the performance to a new state of the art with an 89% accuracy.

II. Related Work

Many teams shared their solutions and discussion on this problem in the Kaggle forum, among which some of the work realized significantly high score ($>$90% accuracy, $<$ 0.15 log loss). The feature engineering seems to be the most important part of this competition since most of the top teams [1] [2] [3] [4] [5] credited their success to important features. There were mainly three types of features we can extract from this dataset: embeddings feature (for neural networks), traditional NLP features (common n-grams, Length of questions, etc.), and structural features (common neighbors of two questions, clique features, etc.), The discussion on Kaggle [1] [17] indicated that the most significant score improvement was given by structural features, followed by a bunch of traditional NLP features. However, although single deep learning models based on embedding features did not perform as well as single traditional models [19], they provided more diverse ways for prediction and could be good tools for feature extraction [18]. Some team also proposed high rank pure neural network solution [6] where they try various type of deep learning architecture and ensemble them together.

III. Method:Feature Engineering

1) feature extracting

In most of the Kaggle competitions, the algorithm is fixed while the feature engineering is always the key to success. In this section, we elaborate on feature engineering, which

includes the extraction of natural language processing features and magic features.

Basic text features:

a. Length information: Character and word length.

b. Length difference information: Character and word level length absolute differences, log length absolute differences, length ratio, log length ratio.

c. Common word intersection count.

d. Sum number of words in two questions.

e. sum total of frequency of two questions.

f. absolute difference of frequency of two questions.

g. Punctuation symbol count.

h. Number in question count.

Fuzzy string features:

This is for string matching and uses Levenshtein distance to calculate the differences between two sequences.

a. Word-to-Mover Distance features: It is useful in cases where two questions have different wordings but convey the same information.

Embedding features:

Word2Vec pretrained on Google News, GloVe. We basically sum all of word embeddings and normalize it to unit vector to represent a sentence.

a. Embedding distance features: We compute various distance metrics, such as cosine, cityblock, canberra, euclidean, minkowski, braycurtis distance.

b. Embedding skewness and kurtosis: Skewness is a measure of lacking of symmetry. Kurtosis is a measure of whether the data are heavy-tailed or light-tailed relative to a normal distribution.

"magic" features:

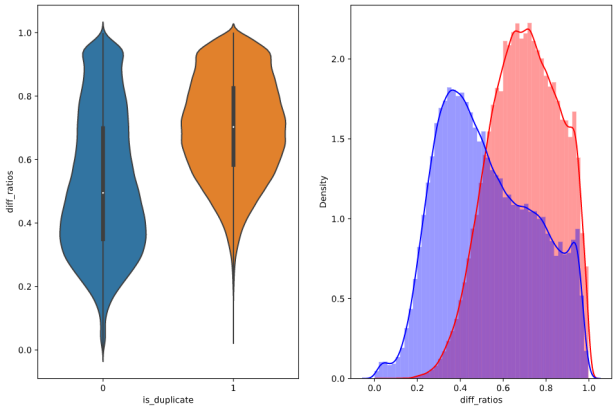
a. Question ID: Originally, qid is only an identifier for specific question. But one of the participants found that the qid is actually a time dependent serial number. Furthermore, the duplicate label ratio becomes lower and lower as the time pass. The hypothesis is that Quora itself also tried to reduce duplicated question in this period of time. So, the qid becomes a strong feature.

b. Numbers of question occurrence: One suggested that the duplicated questions tend to be the same group of questions. In other words, the more time a question appears in the training set, the more probable such question pair is duplicated regardless of what question is paired with it.

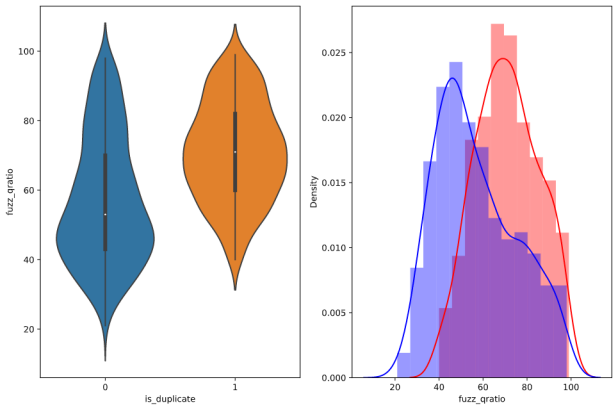
It is worth mention that those magic features end up helping a lot. Though we call those features "magic", in fact, we think those magic features basically capture some hidden phenomenon behind the data, which make them become strong features in this task.

2) feature analysis

A) The distributions for diff_ratios have some overlap on the far right-hand side but the degree of differentiation is quite obvious. For the interval from 0.0 to 0.4, the probability of not duplicate is significantly higher than duplicate. So the classification is clear.

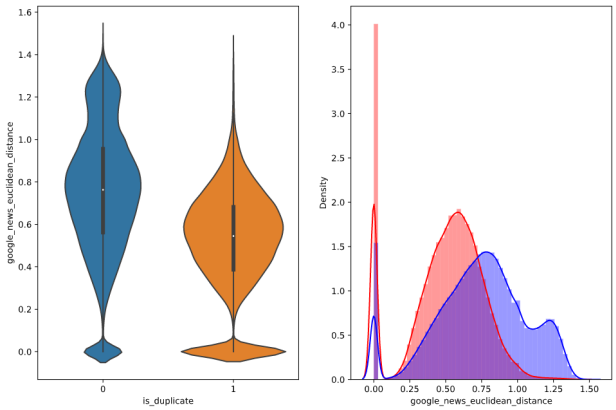


B)



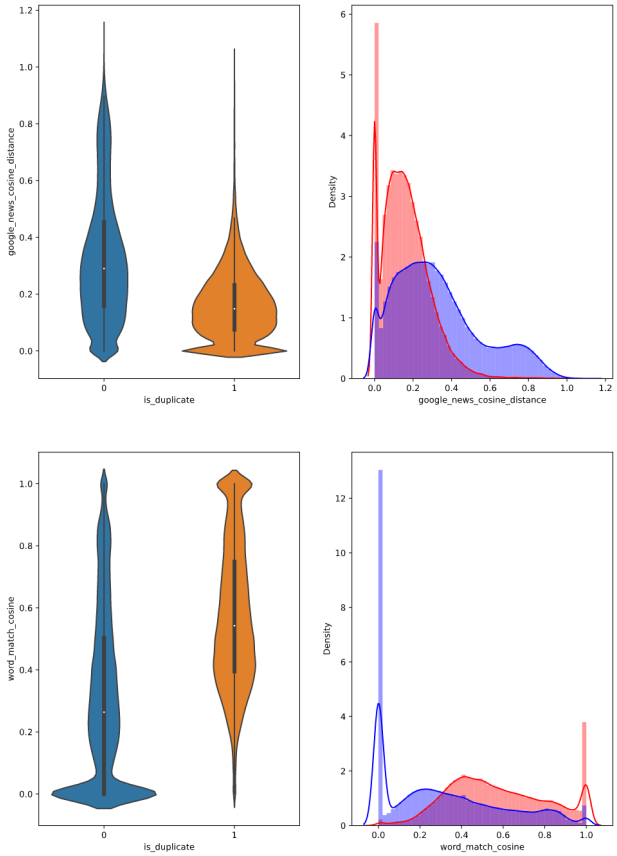
The distributions for fuzz_qration has room for differentiation. The classification has good effect.

C)



The distribution for google_news_cosine_distance and google_news_euclidean_distance are relatively similar for duplicate and nonduplicate. The classification is not clear.

D) The distribution for word_match_cosine is quite separate. The peak of is_duplicate is around 1 while the peak of not duplicate is around 0.



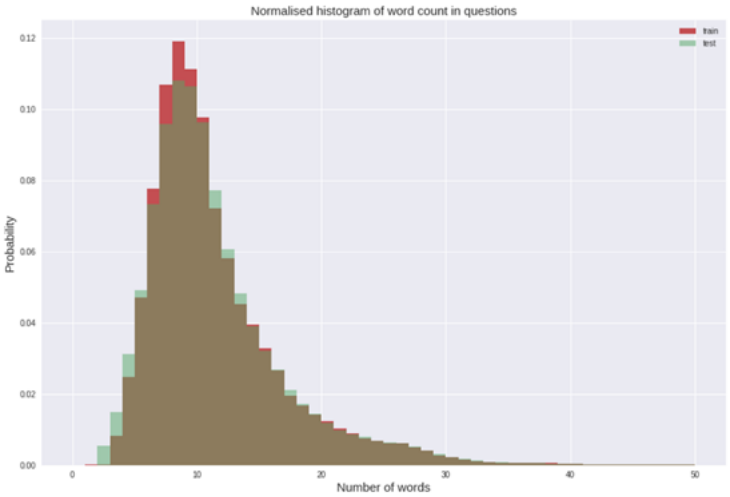


Fig. 1. The number of words in each sentence [11]

IV. METHOD:DEEP LEARNING

A. pre-pocessing for DL

We first swap the place of question1 and question2 for each compared pair to augment the total data set (since swapping the two compared candidates will not change the label). Both the training set and test set are cleaned (we do spell correction, question mark separation) and tokenized. Since the number of words in each sentence is 11.06 on average [Anokas] and almost all of them are below 30 words, we pad each question into length on 30. Then we prepare the Glove embedding matrix based on words appear in test dataset or train dataset where each word has a dense representation of 300 dimension.We first swap the place of question1 and question2 for each compared pair to augment the total data set (since swapping the two compared candidates will not change the label). Both the training set and test set are cleaned (we do spell correction, question mark separation) and tokenized. Since the number of words in each sentence is 11.06 on average [Anokas] and almost all of them are below 30 words [Fig.1], we pad each question into length on 30. Then we prepare the Glove embedding matrix based on words appear in test dataset or train dataset where each word has a dense representation of 300 dimension.

B. MLP

MLP model serves as baseline test for the project, in which the forward pass has 6 dense layers from embedding to classification output, the feature engineering output is concatenated into the network at the second layer of MLP. We use drop out and batch normalization for every dense layer. In the first layer of MLP, since our input dimension is a matrix of words and their embedding vectors, we adopted time distributed dense layers and max pooling layer after that. The max pooling works along the axis of time steps (the axis of different words). so that our output is one dimensional.

C. CNN

Here we present our CNN based model which are composed of the following components: input encoding, 1D convolution sequence, and inference composition. Figure X shows a high-level view of the architecture. Vertically, the figure depicts the three major components, and horizontally, the second layer represents all 8 convolution we applied, and the second layer represents composition of features extracted both from convolution neural network and feature engineering.

1D convolution sequence

We apply 1D convolution as the central block of our CNN based model, the idea of using convolution is to generate extra feature among a general scale of words. The average question length in Kaggle data set is about 15 and we use here kernel size from 1 to 8. In dealing with short-distance dependence, our 1D convolution perform well. With two sentences(a is a premise and b is a hypothesis):

$$a = (a1, \ldots\ldots, a(la))$$

$$b = (b1, \ldots\ldots, b(lb))$$

The ai or bi is l-dimensional embedding vector, which initialized by our pre-trained word embeddings. We first use 1D convolution to encode the a and b. We have:

$$a\hat{Z} = [(a_1\hat{Z}); \ldots\ldots; (a_i\hat{Z})]$$

$$b\hat{Z} = [(b_1\hat{Z}); \ldots\ldots; (b_i\hat{Z})]$$

as the concatenated output of our different kernel, for i is the kernel size and each one has 128 filters. They are all pooling into 1 dimension before concatenation.

Inference composition

In our models, we further enhance the local inference information collected. We compute the difference and the element-wise product between two question and add the traditional feature.

$$m = [a\hat{Z} - b\hat{Z}; a\hat{Z} * b\hat{Z}; t_f]$$

This process could be considered as a special case of combining high order interaction between elements.

D. LSTM

The LSTM refers to Long-Short term memory, which uses a set of gates to determine whether the output hidden states should remember or refresh the information from previous time steps. The LSTM can be expressed as follows.

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

$$c_t = f \odot c_{t-1} + i \odot g$$

$$h_t = tanh(c_t)$$

The LSTM can better grasp previous layer's memory to estimate the overall sentence structure. We added bidirectional feature to the LSTM layer so that it can process the input sequence in reverse order. We think this bidirectional LSTM layer can serve as a feature extractor to gather the information learned from each question. With that feature tensor calculated, we didn't directly apply pooling and concatenate them, we chose to let this LSTM layer return the entire sequence of hidden states as input to attention layer so that we can merge the information of two questions. Then we can feed the output of attention layer to another layer of bidirectional LSTM, at which we only output the final hidden states as our final feature representation.

Suppose we choose to have fixed length N word sentence with d dimensional word embedding, and d' dimension of RNN hidden states. The input tensor to the tensor is in $R^{N\times d}$. Since we have to questions to process, we defined two paths of layers before the fully connected layers. Let W_1 denotes the input tensor of bidirectional LSTM layer for question 1 and W_2 denotes the input tensor for question 2. Since we return the whole sequence at the first bidirectional LSTM, the output shape is $N \times d'$, denote the two tensors as A_1, A_2. The attention layer performs the operation described below.

$$let\ A = A_1 A_2^T = \begin{pmatrix} \vec{a}_1 & \vec{a}_2 & \ldots & \vec{a}_N \end{pmatrix} = \begin{pmatrix} \vec{b}_1^T \\ \vec{b}_2^T \\ \vdots \\ \vec{b}_N^T \end{pmatrix} \in R^{N\times N}$$

$$define\ \vec{\alpha}_i = softmax(\vec{a}_i),\ \vec{\beta}_i = softmax(\vec{b}_i)$$

$$softmax(\vec{v}) = \vec{d}\ where\ d_i = \frac{exp(v_i - max(\vec{v}))}{\sum exp(v_j - max(\vec{v}))}$$

$$A_1' = \begin{pmatrix} \vec{\beta}_1^T A_2 \\ \vdots \\ \vec{\beta}_N^T A_2 \end{pmatrix} \in R^{N\times d'}$$

$$A_2' = \begin{pmatrix} \vec{\alpha}_1^T A_1 \\ \vdots \\ \vec{\alpha}_N^T A_1 \end{pmatrix} \in R^{N\times d'}$$

The attention output makes a mixture of information between question 1 and question 2. Tensor A_1' and A_2' are combined with A_1, A_2 to fed into second layer of bidirectional LSTM, where the attention information extracted can be further mixed up. The way of combining attention output and attention layer input is as follows, and we denote the input tensor to second bidirectional LSTM layer as W_1', W_2'.

$$W_1' = [A_1, A_1', A_1 - A_1', A1 \odot A1'] \in R^{N\times 4d'}$$

$$W_2' = [A_2, A_2', A_2 - A_2', A2 \odot A2'] \in R^{N\times 4d'}$$

The output of the second LSTM is only the final hidden state $\vec{h}_N \in R^{d'}$. The hidden state of two questions and also the feature engineering output are concatenated for the latter fully connected layers as a classifier.

V. METHOD: TRADITIONAL MODELS

Tree boosting is an optimized distributed gradient enhancement library designed to be efficient, flexible and portable. It is very famous among Kagglers [20].

VI. METHOD: MODEL ENSEMBLE

Most of teams found that in this competition, diversity was more important than individual model accuracy (include worse but more diverse model into the). We performed a 2-layer model stacking. In the first layer we used our LSTM based model (trained on raw data), our 1D-CNN model (trained on raw data), one XGboost model (trained on features extracted by tradition methods), one MLP model (trained on raw data and concatenated with traditional features on later layers), our 1D-CNN model (trained on raw data and concatenated with traditional features on later layers), our LSTM model (trained on raw data and concatenated with traditional features on later layers). And in the second layer we use another XGboost model. We use 5-fold cross validation to reduce overfitting problem. Although our single neural network models perform worse than single traditional models, the result shows that our models introduce more diversity to the stacked final model, gaining 0.02 more score in return.

VII. EXPERIMENT

We make a 9:1 split on the given training set (404289 samples in total, 808578 samples after data augmentation) for training and validation. Below, we list our training details. We use the nAdam method for LSTM's optimization and Adam for the rest of the models (with default momentum in Keras). All word embeddings have 300 dimensions. All sentences are padded to 30 words. All models are trained with batch size 1024. The metrics for saving the best model parameter is accuracy. We use 5-fold cross validation for the model stacking part.

Pre-trained 300-D Glove 840B vectors was used to initialize our word embeddings. For CNN model, we use 0.2 dropout rate for layers before the pooling layer, and 0.3 dropout rate for dense layers after that, the l2-weight-decay is set to 0.0001. For MLP model, we use 0.2 dropout rate feature layers and 0.1 for all other layers, the l2-weight-decay is set to 0.0001. For LSTM model, we use 0.25 dropout rate for LSTM layers, 0.1 dropout for pooling layers, 0.3 dropout for dense layers, the l2-weight-decay is set to 0.0001.

We also released our code for reproducing our experiment.

VIII. RESULT

A. performance

TABLE I

model	*Loss*	*Accuracy*	*Loss**	*Accuracy**
MLP	***0.46***	***0.81***	***0.23***	***0.891***
CNN	***0.37***	***0.82***	***0.24***	***0.893***
Enhanced LSTM	***0.36***	***0.815***	***0.23***	***0.895***
xgboost	***0.32***	***0.86***	***0.17***	***0.925***
Final model:stacking	***None***	***None***	***0.15***	***0.931***

[a] * means adding traditional features to input

Our best score achieved rank 279 (bronze medal) among 3296 competitors in this Kaggle challenge.

B. analysis

Why deep models did not perform well in this task? We think it is probably because the input question are generally short questions, and the model just need to focus on some keywords to determine whether the two questions are the same. For short Quora questions, the overall sentence structure is not important. According to the experimental result, the LSTM model did not have significant superiority over simple MLP model. Since both the LSTM and CNN architecture can be regarded as feature extractors from raw sentences, for latter fully connected layers to classify, if these features extracted didn't help improve the performance, it could be deduced that the optimization process were lowering the weights of features learned from previous LSTM or CNN layers, but rather putting much more weights on the features specifically collected by traditional algorithms. For the model, the traditional features were easier to learn, so our CNN, LSTM models might finally get reduced to MLP model. By our observation, the complicated models started to overfit at the same validation accuracy with the simple models, so it is also possible that the training set is not large enough, so both MLP and complicated models like LSTM and CNN started to overfit at a validation accuracy around 89%.

IX. CONCLUSION

We proposed both traditional machine learning and neural network method in this task and achieved much better performance by feature engineering. Neural network's performance didn't reach our expectation because they cannot extract powerful enough features like those "magic" features. We also applied a committee mechanism and further improved the overall accuracy, suggesting the importance of model diversity over complexity. Future work we might do can be improving the performance of neural network based model. One approach to optimize in training time is to treat LSTM and CNN as feature extractor and training their parameters independently, then we can apply transfer learning with the parameters of LSTM and CNN frozen, then train the rest of network.

X. CONTRIBUTION

Contribution:
Zhongbo Zhu: LSTM,xgboost code writing
Bo Pan: model integration
Shuhong Xiao: CNN code writing
Xinyu Lian: traditional features extraction
Yicheng Lu: model integration

REFERENCES

[1] Maximilien, "First place solusion," kaggle.com, 2017. https://www.kaggle.com/c/quora-question-pairs/discussion/34355 (accessed May 27, 2021).

[2] Silogram, "Overview of 2nd-Place Solution," kaggle.com, 2017. https://www.kaggle.com/c/quora-question-pairs/discussion/34310 (accessed May 27, 2021).

[3] J. Turkewitz, "Overview Of 3rd Place Solution," kaggle.com, 2017. https://www.kaggle.com/c/quora-question-pairs/discussion/34288 (accessed May 27, 2021).

[4] HouJP, "Overview of 4th-Place Solution," kaggle.com, 2017. https://www.kaggle.com/c/quora-question-pairs/discussion/34349 (accessed May 27, 2021).

[5] Faron, "5th Place Solution Summary," kaggle.com, 2017. https://www.kaggle.com/c/quora-question-pairs/discussion/34359 (accessed May 27, 2021).

[6] Aphex, "7-th solution overview," kaggle.com, 2017. https://www.kaggle.com/c/quora-question-pairs/discussion/34697 (accessed May 27, 2021).

[7] "8th solution with part of source code(under construction)," kaggle.com, 2017. https://www.kaggle.com/c/quora-question-pairs/discussion/34371 (accessed May 27, 2021).

[8] CPMP, "Solution 12 overview," kaggle.com, 2017. https://www.kaggle.com/c/quora-question-pairs/discussion/34342 (accessed May 27, 2021).

[9] Alex, "1D CNN (single model score: 0.14, 0.16 or 0.23)," kaggle.com, 2017. https://www.kaggle.com/rethfro/1d-cnn-single-model-score-0-14-0-16-or-0-23 (accessed May 27, 2021).

[10] E. Ahmet, "24th Place Solution Repo," kaggle.com, 2017. https://www.kaggle.com/c/quora-question-pairs/discussion/34534 (accessed May 27, 2021).

[11] Anokas, "Data Analysis XGBoost Starter (0.35460 LB)," kaggle.com, 2017. https://www.kaggle.com/anokas/data-analysis-xgboost-starter-0-35460-lb (accessed May 27, 2021).

[12] S. R. Bowman, J. Gauthier, A. Rastogi, R. Gupta, C. D. Manning, and C. Potts, "A Fast Unified Model for Parsing and Sentence Understanding," in Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Aug. 2016, pp. 1466–1477, doi: 10.18653/v1/P16-1139.
[13] A. Graves, Supervised Sequence Labelling with Recurrent Neural Networks, vol. 385. Stud Comput Intell, 2012.
[14] Kai Sheng Tai, R. Socher, and C. D. Manning, "Improved semantic representations from tree-structured long short-term memory networks," 2015.
[15] L. Mou et al., "Natural language inference by tree-based convolution and heuristic matching," 2016.
[16] Q. Chen, X. Zhu, Z.-H. Ling, S. Wei, H. Jiang, and D. Inkpen, "Enhanced LSTM for natural language inference," Jul. 2017, pp. 1657–1668, doi: 10.18653/v1/P17-1152.
[17] D. Thaler, "Train/Test Differences," kaggle.com, 2017. https://www.kaggle.com/c/quora-question-pairs/discussion/34291 (accessed May 27, 2021).
[18] Z. Wang, "zhaoyu18/quora-question-pairs," GitHub, May 17, 2021. https://github.com/zhaoyu18/quora-question-pairs (accessed May 27, 2021).
[19] A. Massiot, "Deep Learning did not win ?," kaggle.com, 2017. https://www.kaggle.com/c/quora-question-pairs/discussion/34329 (accessed May 27, 2021).
[20] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," Jun. 2016. [Online]. Available: https://arxiv.org/pdf/1603.02754.pdf.