Quora Duplicate Questions Identification

Chen Xia (chenxia) **Xuanyue Yang** (xuanyuey) **Wenting Ye** (wye2)
Carnegie Mellon University
5000 Forbes Ave, Pittsburgh, PA 15213

Abstract

In this report, we investigate two different methods to identify the duplicate question in Quora dataset, which are deep neural network and Gradient Boosting Decision Tree (GBDT). The first network adopts the Siamese architecture with ELMo and Convolutional Neural Network (CNN) as feature encoder. The embedding of two sentences as well as their distance and angle were fed into fully-connected layer for prediction. The second model uses 105-dimension handcrafted feature including information regarding the length, semantic embedding, similarity and TF-IDF for identification. Our best model achieves 89.2% accuracy and 0.961 AUROC on the test set.

1 Introduction

Recognizing textual entailment (RTE) is concerned with determining the relationship between two text fragments. This requires a model to understand the semantic information conveyed in the two text. In this project, we investigate a task similar to RTE, which aims to identify the duplicate Quora questions [2]. This dataset is very important in reality because duplicate question, such as "What is the most populous state in the USA?" and "Which state in the United States has the most people?", should not co-exist because they share the same logical meaning.

2 Related work

In traditional machine learning, short text matching often is tackled by random forest [1], gradient boosting decision tree [3]. Random forest are an ensemble learning method constructing a multitude of decision trees (each tree is trained using partial data) and predict using the mode of the output (for classification). GBDT is also an ensemble method but each decision tree is to fit the gradient of the current time step loss.

Recent years has witnessed the success of the deep neural network especially recurrent neural network in natural language understanding. [11] proposes a mutual attention mechanism. In each matching direction, each time step of one sentence is matched against all timesteps of the other sentence from multiple perspectives. Their model achieves state-of-the-art result with accuracy 88.17.

3 Model

3.1 End-to-End Neural Network

We first explore using a end-to-end neural network to encode the question and do binary classification. Figure 1 shows our main model architecture. We adopt ELMo [8] as our text encoder which incorporates extra contextual information than other common word embeddings through biLSTM [10], and addresses the out-of-vocabulary word problem through character-level CNN [6]. We then use a multi-layer CNN to extract sentence features based on the ELMo-encoded sequence, and obtain a fixed size embedding with average pooling for each of the two question text. To make binary

prediction whether the question pair is duplicate, we concatenate the two embedding along with their L_2 distance and cosine similarity as the feature vector, and pass it through a dense MLP.

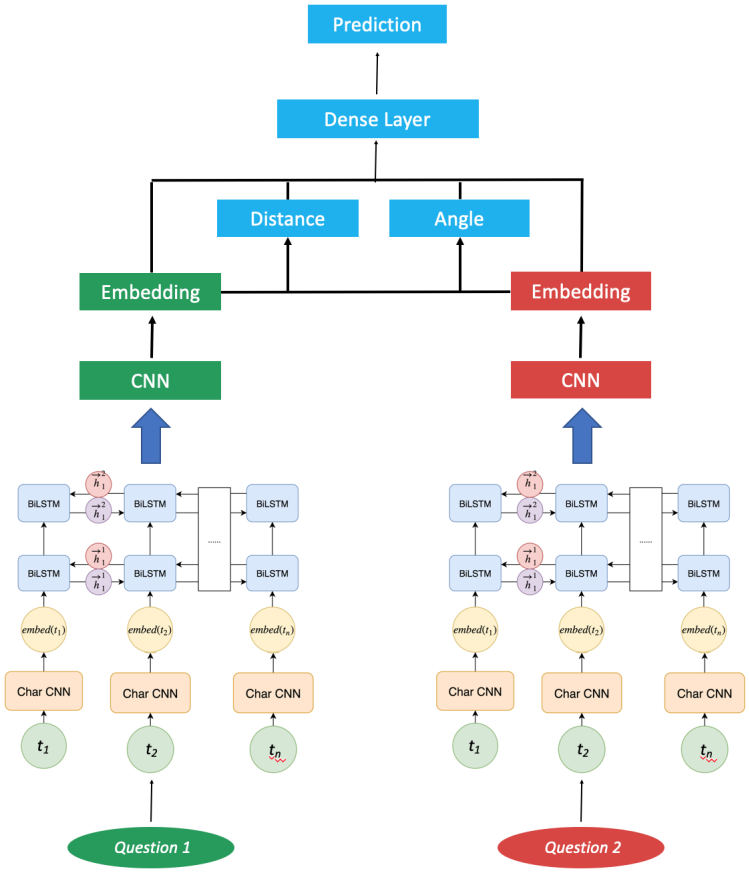

Figure 1: ELMo + CNN Model Architecture

3.2 GBDT with Hand-crafted Features

From traditional machine learning's perspective, we need to do first feature engineering for the textual question pairs, and then train a binary classifier based on the feature representations. We use Gradient Boosting Decistion Trees (GBDT) [3] as our feature-based model since as a effective boosting method, it has strong non-linear fitting capability, achieving state-of-the-art results among in many traditional machine learning tasks.

GBDT is based on a set of K regression decision trees:

$$f_k(\mathbf{x}) = \sum_{k}^{K} g_k(\mathbf{x}), \tag{1}$$

with each tree g_{k+1} fitting the negative gradient of the loss function:

$$y_{k+1}^{(i)} = -\lambda \frac{\partial \mathcal{L}(y_0^{(i)}, f_k(\mathbf{x}^{(i)}))}{\partial f_k(\mathbf{x}^{(i)})}, \tag{2}$$

where $y_{k+1}^{(i)}$ is the target for $(k+1)$-th tree g_{k+1} with input of the i-th instance $\mathbf{x}^{(i)}$, and $y_0^{(i)}$ is the original target in the dataset. I.e., each new decision tree does the gradient descent based on the loss function calculated with current set of trees in the function space. It benefits from decision trees such that it does not require the normalization of the input, and it could prevent overfitting to some extend with the help of boosting.

4 Experiment

4.1 Dataset

The Quora duplicate question pair dataset contains 404,363 pairs of question pair. Among them, 255k pairs of question are identify as duplicate while the other 149k pairs are normal.

Preprocessing We drop all instances with null question (4 in all). We then tokenize the questions using `Stanford CoreNLP` tokenizer. For our character-level encoder model, we convert all non-ascii characters to their unicode hex values.

Partitioning We partition the dataset randomly using ratio $7:1:2$ as our training, validating and testing set respectively.

4.2 Feature Engineering

4.2.1 Text Features

The all text feature we use could be categorized into following 4 groups:

1. Length feature: char length, char length difference, char length ratio, word length, word length difference, word length ratio, average word length, average word length difference, number of capitalized words
2. Embedding feature (glove, google news, fast text [4]): cosine/euclidean/jaccard/minkowski distance,word mover's distance
3. Similarity feature: word/word bigrams/word trigrams distance, char bigrams/char trigrams distance
4. TF-IDF feature: TF-IDF mean/sum/length, char counter vector unigram/bigram/trigram jaccard/euclidean distance, char out of fold feature, char svd, char svd out of fold feature, word euclidean distance, word out of fold feature

4.2.2 Graph Features

After a closer scrutiny of the dataset, we observe that:

1. The higher the frequency of the problem in the data set, the greater the probability of repeating the problem.
2. The more common neighbors the question pair shares, the greater the probability that they are duplicated.

Hence, we could create graph which depicts the relationship between questions. For each pair of questions in the dataset, we create an edge between them. Then based on this graph, we design some features like **graph clique** [9] and **graph page rank** [7]

4.3 Experimental Settings

For the GBDT model we use `lightGBM` [5] toolkits from Mircrosoft, and the configuration is shown in Table 1. For the neural network model we use `PyTorch` and `AllenNLP`, and the configuration is shown in Table 2.

4.4 Results

4.4.1 Neural network

The deep neural network achieve 82.7% accuracy and AUROC of 0.900. If we calculate the feature without final MLPs and train a GBDT on it, we could improve the accuracy by 0.8%, which shows GBDT could be a better classifier than MLP. However, the result is still not comparable with the human-crafted feature, which means the feature extract by ELMo and CNN is worse than human-crafted feature with informative prior knowledge in this task.

Boosting Type	**Objective**	**Metric**	**Number of Leaves**
gbdt	binary	auc, log loss	256
Learning Rate	**Feature Fraction**	**Bagging Fraction**	**Bagging Frequency**
0.02	0.8	0.8	5

Table 1: LightGBM Training Settings

ELMo Representation Dimension	**CNN**	**MLP**
256	Kernel Width 3, 3 Layers	3 Layers
Dropout Probability	**Learning Rate**	**Optimizer**
0.2	$8 \cdot 10^{-4}$	Adam

Table 2: Neural Network Settings

4.4.2 GBDT

To investigate the importance of features, we first train the GBDT with only 10 basic features, resulting a model with 73.6%. As shown in Table 3, the performance could be significantly improved when increasing the number of features. GBDT with 102-Dim feature and 3-Dim graph feature achieves the best result with 89.2% accuracy and 0.961 AUROC.

We also conduct a ablation experiment where we train a model without graph features. The training accuracy decreases by 1% in Table 3, which shows that graph features expresses the direct relationship between question pairs and gives a lot of prior knowledge we need in training the model.

Finally, we evaluate how sensible GBDT are to the size of training data. As shown in Table 4, we reduce the number of training data by 90% each time. However, the accuracy of GBDT only decreases by about 1%, which shows that GBDT could behave well in limited-size dataset. Due to resource and time, we are not able to conduct the same experiment for neural-based model. But we expect the neural network requires large number of sample to train.

5 Conclusion

In conclusion, we both use traditional statistical models and neural nets to try to address this problem. In the GBDT method, we extract some features related to not only using nlp technique, but also exploring the graph feature which are potential features. In neural network methods, we using CNN to automatically extract features and use DNN to do the classification. Both models give somewhat good results. But the GBDT scores better, and is more robust.

6 Future work

There are still some problems needed to be addressed. First, each text includes some noisy texts, like "Japanese" in English. We could first extract these part of texts, and then use machine translation methods to translation Japanese into English, then compare the two "English sentences". Second, there are still some semantic meanings underlying each question pair. So we could further extract semantic features from the dataset, and it will give more potential features just like graph features.

Model	**Log Loss**	**Accuracy (%)**	**AUROC**
10D HCF + GBDT	0.477	73.6	0.820
102D HCF + GBDT	0.252	88.2	0.955
102D HCF + 3D Graph Feature + GBDT	**0.238**	**89.2**	**0.961**
ELMo + CNN + Distance/Angle + MLP	0.458	82.7	0.900
ELMo + CNN + Distance/Angle + GBDT	0.369	83.5	0.912

Table 3: Performances under different settings

Percentage of training data	Log Loss	Accuracy (%)	AUROC
1	0.238	89.2	0.961
0.1	0.269	87.9	0.949
0.01	0.307	86.2	0.934
0.001	0.371	83.2	0.907

Table 4: GBDT Performance under different number of training data

References

[1] L. BREIMAN, *Random forests*, Machine learning, 45 (2001), pp. 5–32.

[2] Z. CHEN, H. ZHANG, X. ZHANG, AND L. ZHAO, *Quora question pairs*, 2018.

[3] J. H. FRIEDMAN, *Greedy function approximation: a gradient boosting machine*, Annals of statistics, (2001), pp. 1189–1232.

[4] E. GRAVE, T. MIKOLOV, A. JOULIN, AND P. BOJANOWSKI, *Bag of tricks for efficient text classification*, in Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL, 2017, pp. 3–7.

[5] G. KE, Q. MENG, T. FINLEY, T. WANG, W. CHEN, W. MA, Q. YE, AND T.-Y. LIU, *Lightgbm: A highly efficient gradient boosting decision tree*, in Advances in Neural Information Processing Systems, 2017, pp. 3146–3154.

[6] Y. KIM, *Convolutional neural networks for sentence classification*, arXiv preprint arXiv:1408.5882, (2014).

[7] L. PAGE, S. BRIN, R. MOTWANI, AND T. WINOGRAD, *The pagerank citation ranking: Bringing order to the web.*, tech. rep., Stanford InfoLab, 1999.

[8] M. E. PETERS, M. NEUMANN, M. IYYER, M. GARDNER, C. CLARK, K. LEE, AND L. ZETTLEMOYER, *Deep contextualized word representations*, arXiv preprint arXiv:1802.05365, (2018).

[9] R. A. ROSSI, D. F. GLEICH, A. H. GEBREMEDHIN, AND M. M. A. PATWARY, *Fast maximum clique algorithms for large graphs*, in Proceedings of the 23rd International Conference on World Wide Web, ACM, 2014, pp. 365–366.

[10] M. SCHUSTER AND K. K. PALIWAL, *Bidirectional recurrent neural networks*, IEEE Transactions on Signal Processing, 45 (1997), pp. 2673–2681.

[11] Z. WANG, W. HAMZA, AND R. FLORIAN, *Bilateral multi-perspective matching for natural language sentences*, arXiv preprint arXiv:1702.03814, (2017).