Duplicate Questions Identification: Exploring Effects of Feature Representation on Deep Neural Nets and Random Forest Models

Fan Fan, Hanyang Li, Jiazhi Zhang
Department of Machine Learning
Carnegie Mellon University
Pittsburgh, PA 15213
{ffan2, hanyangl, jiazhiz}@andrew.cmu.edu
Andrew IDs: ffan2, hanyangl, jiazhiz

Abstract

In this project, we aim to tackle the problem of identifying duplicate questions on Quora. We propose two closely-related methods: deep neural networks and random forests. We first train LSTM networks to learn an appropriate vector representation for each question. From these representations, we further employ methods of feature engineering to boost the performance of both methods. Our experiments show that a task-specific representation can be extremely helpful and additional feature engineering upon those representations can make further improvements on classification. Both of our methods achieve an accuracy over 85%.

1 Introduction

Nowadays, Quora has become one of the largest platform to ask questions and share knowledge. However, the existence of multiple questions with same intent hinders people from efficiently finding the best answers to their questions. Calling for a good solution for this problem, Quora released a data set to seek methods that can identify duplicate questions. This data set contains over 400,000 pairs of questions, each labeled as 0 (not duplicate) and 1 (duplicate).

An important first step of identifying duplicate questions is to properly represent sentences as feature vectors. It is widely known that intermediate signals of neural networks forward pass before the last fully-connected layers can be used as vectors for sentence representation. Many techniques have been proposed to extract sentence feature representation using neural networks and many neural network classifiers have been proposed to make classification based on this learned representation Jiang et al. [2017] Tai et al. [2015]. However, how the sentence representation learned by neural networks affect the performance of non-neural network models still remains an open question.

In this project, we make the hypothesis that properly learned, task-specific sentence representation can boost the performance of random forest models in duplicate question identification. The objective of our project is to test this hypothesis through two steps. First, we design different strategies to learn proper sentence representations using neural nets. The higher accuracy a neural net based classifier can achieve with a certain representation, the better we consider the representation is. Next, we apply the good representations learned to a random forest model and test if they significantly improve the classification performance of the random forest model.

32nd Conference on Neural Information Processing Systems (NIPS 2018), Montréal, Canada.

2 Related Works

2.1 Quora Random Forest

The Quora Engineering Team proposed a random forest model (Jiang et al. [2017]) which makes classification based on a set of handcrafted feature, including the number of common words, the number of common topics labeled on the questions, and the part-of-speech tags of the words. Quora did not provide further details of this model, but it inspired us to design our random forest model, which we will discuss in the method section.

2.2 Quora LSTM Model

The same team also proposed a Recurrent Neural Network (RNN) model Tai et al. [2015] to approach the duplicate detection challenge. This is an end-to-end solution so there is no handcrafted feature involved. In this method, the team first trained their own word embeddings using Quora's text corpus. Then, they used the Long Short Term Memory network (LSTM) variant of RNN to extract sentence feature embeddings for the two sentences. The reason for using LSTMs is that they are good at capturing long-term dependencies in sequential data Hochreiter and Schmidhuber [1997] . Then, the team concatenated embeddings of the two questions and fed the concatenated representation into a dense layer to make a final classification. In further exploration, the team also attempted to use the distance (the sum of squares of the difference of the two vectors) and the angle (an element-wise multiplication of the two vector representations) of the two question representation to make final classification. These feature representation strategies inspired our model design.

3 Dataset and Preprocessing

Dataset: The dataset used is the Question Pairs Dataset from Quora. It contains 40,4351 question pairs. Each pair is labeled as either 1 (duplicate) or 0 (not duplicate). In the dataset, 25,5045 (63.1%) examples are duplicate.

Preprocessing: In this project, we use GloVe (by Pennington et al. [2014]) as our pretrained word vector representation. GloVe is a word-to-vector mapping unsupervisedly learned from predetermined dataset. The particular version of GloVe we used is learned from Wikipedia 2014 dataset and Gigaword 5 dataset. For our random forest models, we use GloVe to convert words in original questions to vector representations after we tokenize the lowercase raw data, and use those as input. For neural network models, we define an embedding layer for each model and initialize its weight according to GloVe mapping.

4 Methods

4.1 Neural Networks

Variations of deep neural networks, especially LSTM models, have shown significant potential in modeling sequential data. LSTM models use memory cells to store information over a span of time steps. The underlying logic of LSTM is illustrated by the following equations, with *h* being hidden state and *c* being cell state and sigm and $\tanh$ being applied element-wise:

$$
\begin{aligned}
&h_t^{l-1}, h_{t-1}^l, c_{t-1}^l \to h_t^l, c_t^l \\
&\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \mathrm{sigm} \\ \mathrm{sigm} \\ \mathrm{sigm} \\ \tanh \end{pmatrix} T_{2n,4n} \begin{pmatrix} h_t^{l-1} \\ h_{t-1}^l \end{pmatrix} \\
&c_t^l = f \odot c_{t-1}^l + i \odot g \\
&h_t^l = o \odot \tanh(c_t^l)
\end{aligned}
$$

In this project, we use bidirectional LSTM (Graves et al. [2005]) as our baseline, then we augment our base model with a CNN feature extraction layer and a distance-angle operation to improve embedding quality.

Baseline - BiLSTM: We use the hidden state of a bidirectional LSTM as question embedding for each question and directly concatenate the pair of question embeddings. We then feed the embeddings through one fully-connected(fc) layer before the final output prediction layer, where we use tanh as the irregularity for our fc layers. We set the BiLSTM hidden dimension to 150 and output dimension to 300. The two fully-connected layers reduce the concatenated vector to a binary class label. The hidden size of the fc layer is set to be 300.

BiLSTM + dist_angle: We discover that the performance of the BiLSTM model when using direct concatenation of LSTM output is not satisfactory as fc layers struggle to learn the difference between the first and the second half of the concatenated vector. To solve this issue, We follow the approach presented by this Quora Article Jiang et al. [2017] and instead use absolute value of difference between to question embeddings (distance) as well as the element-wise multiplication (angle) of two question embeddings as the input of fc layers. In this model, the input vector length of fc layer is doubled as it not only includes the original LSTM embedding vector (300×2) but also the distance vector and angle vector (300×2). We will thereby refer to this feature representation as dist-angle representation.

CNN + dist_angle: This model variation maintains the same padding logic and fully-connected layer structure as as BiLSTM + dist_angle while replacing the feature learning structure by CNN (Krizhevsky et al. [2012]). We use Conv1d module with both input and output channel size 300 and kernel size 3.

BiLSTM + CNN: This variation uses the direct concatenation of intermediate outputs from BiLSTM and CNN along with their respective distance-angle vectors as the input of the final classification layers.

BiLSTM + CNN with more feature: In addition to distance and angle as augmented feature vectors, we add element-wise max and min of vectors as extra features. Moreover, we run question-pair embedding vector through fc layers twice, once with switched question order, and calculated the mean of the final output. This additional operation is equivalent to augmenting original data directly by flipping the question pair but is computational more efficient. Augmented with longer vector and data augmentation, this model achieved best accuracy on test set. The architecture of Figure 1

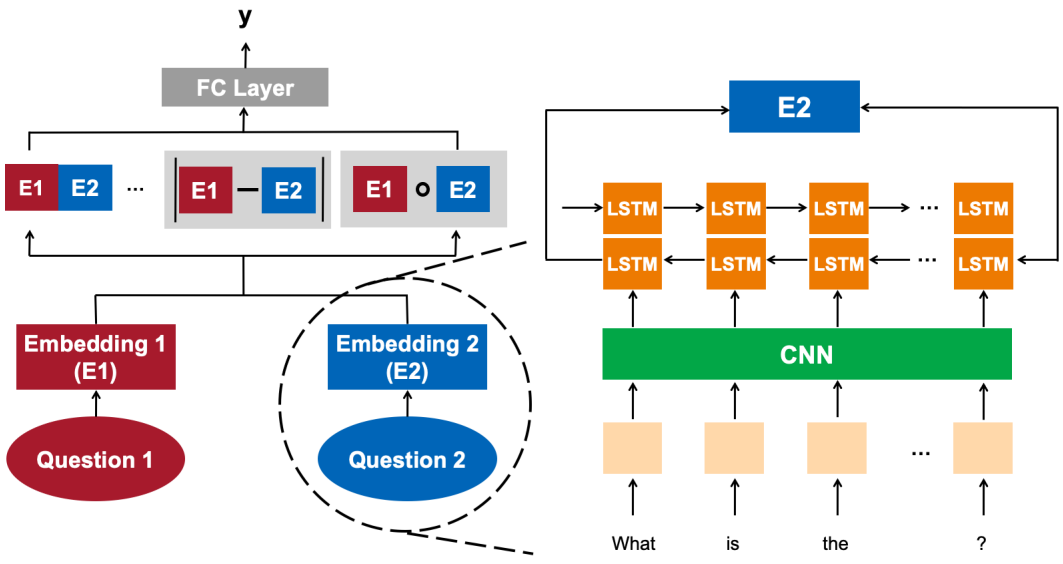

Figure 1: CNN and BiLSTM network for representation extraction and classification

4.2 Random Forest

Quora Random Forest: In our Random Forest approach, we started with Quora Engineer's initial method Jiang et al. [2017] where they constructed a random forest model Breiman [2001] with tens of handcrafted features. However, the article only stated a rough implementation of their proposed random forest model. They only had a few examples such as the number of common words, cosine similarity of the average of word2vec embeddings of tokens, etc. Based on Quora Engineers' descriptions and a similar approach posted by Abhishek Thakur Thakur [2017], we collect a total of 29 handcrafted features consisting of four general sets. The first set is all basic features including the length of both questions, the length difference, number of common words, etc. The second set includes different distance metric between the average of word2vec embeddings of both questions. We use GloVe Pennington et al. [2014] as a pre-trained model to generate an embedding vector for each token and then we take the average for each question. Some examples of distance metric are cosine similarity, euclidean distance, etc. The third set includes features related to Levenshtein distance or edit distance such as Q ratio, token set sort ratio. The fourth set is an extra set based on two sentence embeddings. We have the fourth standardized moment and skewness of both embedding vectors in the set. Furthermore, we include the word2vec embeddings of both questions in the feature set as well. In summary, we gather a total of 629 features for each question pair in the dataset. Table 1 gives an example of a questions pair and the 29 features learned.

There are not a lot of hyperparameters in a random forest model. Our baseline implementation uses a forest of 400 estimators and each level random samples the square root of total features.

Table 1: An example of the 29 handcrafted features for random forest models

Question 1: on snapchat, what happens when you block someone?			
Question 2: if i block someone on snapchat will they still see the last message i sent?			
Label: 0 (not duplicate)			
Basic Features		Distance Metrics	
Feature	Value	Feature	Value
length of q1	49.0	cosine similarity	0.448
length of q2	75.0	euclidean	3.98
different length	-26.0	jaccard	1.0
number of distinct chars q1	19.0	cityblock	23.8
number of distinct chars q2	20.0	canberra	32.4
number of tokens q1	10.0	minkowski	2.37
number of tokens q2	16.0	braycurtis	0.622
number of common tags	5.0		
number of common tokens	5.0		
Levenshtein Features		Embedding-based Features	
Feature	Value	Feature	Value
Levenshtein of questions	53.0	q1 kurtosis	1.32
Levenshtein of part-of-speech tags	32.0	q2 kurtosis	0.192
q ratio	43.0	q1 skew	0.355
wr ratio	86.0	q2 skew	-0.333
partial ratio	53.0		
token set ratio	69.0		
token sort ratio	60.0		
partial token set ratio	100.0		
partial token sort ratio	62.0		

Random Forest Incorporating Learned Features:

In our improved random forest model, there is no major structural change from the baseline. However, in our baseline model, the word2vec embeddings Mikolov et al. [2013] are extracted from GloVePennington et al. [2014] pretrained embeddings from Wikipedia 2014 and Gigaword 5 while this model uses the Vanilla LSTM from to acquire a fixed length (300) embedding for each question. Given these learned embeddings from Vanilla LSTM, we recompute the second and fourth set of

handcrafted features. In order to compare the performance, we keep the same hyperparameters of the Random Forest architecture: 400 estimators and square root of total number of features at each tree level. We also add the two embeddings into the feature vector. In summary, for each questions pair, we gather a total of 629 features and feed them to the Random Forest model. Our model using learned representations instead of pre-trained representations is able to achieve 0.8414 accuracy on the test set.

Random Forest with Learned Features Engineering: To further explore influence of feature engineering on random forest, we first reduce the dimension of LSTM embeddings from 300 to 50 to avoid redundancy in the following feature computations. Then, we adopt the same feature engineering techniques as in the enhanced LSTM model. Given these two embeddings of length 50, besides recomputing the second and fourth feature set mentioned above, we follow the distance and angle computation in terms of of the absolution difference and the element-wise product between two embedding vectors. In this case, we only have a total of 229 features, which is a big reduction from 629 features as in previous settings. Figure 2 shows the input features and architecture of this model.

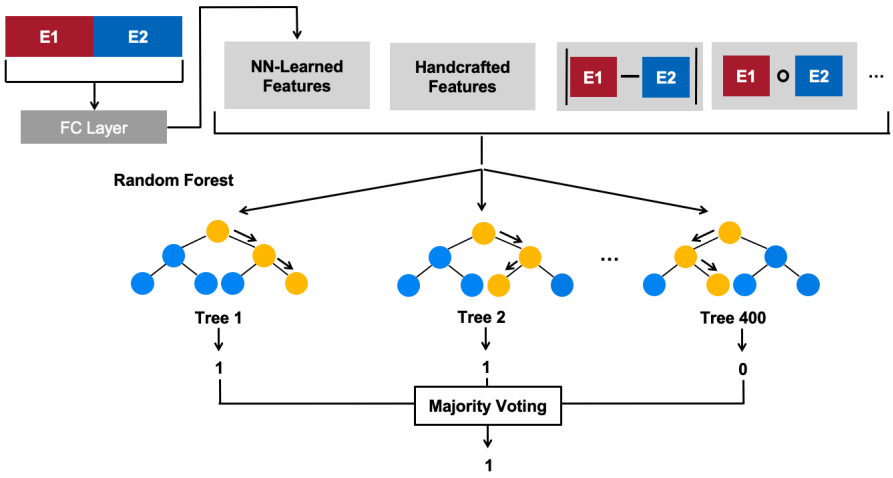

Figure 2: Random forest classifier with learned representation and further feature engineering

5 Results

The results of all our experiments are shown in Table 2. For each individual model, we computes four metrics, including accuracy, precision, recall, and F1 score. For models other than baseline, we evaluate the accuracy improvement compared to their corresponding baseline model.

From the results of deep neural nets, we observe that both architectural changes and feature engineering could potentially improve the expressive power of the network. It can be shown that the angle and distance vectors suggested by Quora Engineering Team captured the relationship of the question pair very well. With the addition of those vectors, our BiLSTM model had the most significant performance improvement. Furthermore, CNNs display fair capacity for extract underlying features and give another performance boost when concatenated with Vanilla LSTM. Finally, when we take into consideration the order of the question pair and add more features such as min and max of two question embedding, we get the best performance out of all the neural network models, These empirical results illustrate the importance of feature engineering. In addition, the incremental improvements with extra feature as well as augmented data (by pair order flipping) demonstrate the capability of neural network in dealing with large scale, high dimension training data but we risk overfitting the dataset. The problem of overfitting could be alleviated by cross validation.

For random forest classifiers, the baseline performance is similar to baseline neural networks. Once we stop using GloVe pre-trained embeddings and start using embeddings from our LSTM models, which are also trained to classify same task, the random forest model gain a significant improvement on the accuracy. This can be explained by the fact that our LSTM models accommodate the embeddings to fit

the task. However, this can also be a potential drawback on our model because the embeddings trained by LSTM networks are biased towards this specific dataset, making our approach less generalizable. Apart from domain specificity, another improvement in our final random forest model is that by reducing the dimension of each question embeddings and adding feature engineering approach, we also gain a strong improvement in terms of test accuracy. This empirical results illustrate the importance of feature engineering and dimension reduction. By reducing embedding dimension from 300 to 50 without hurting testing accuracy, we hypothesize that there are some redundancy in the representations learned by LSTM networks. Removing redundancies and finding concise representations remain a ongoing task. Last but not least, since our final random forest model are composite of two separate methods sequentially, time complexity could be another concern of this approach as our random forest model does not exploit the relatively fast runtime of a general random forest.

Table 2: Accuracy, precision, recall, F1 score, and improvement on accuracy of all models

	Accuracy	Precision	Recall	F1 Score	Improvement
NN Baseline	0.814	0.769	0.716	0.741	
CNN+dist_angle	0.852	0.813	0.780	0.769	+4.76%
BiLSTM+dist_angle	0.856	0.802	0.814	0.808	+5.20%
CNN+BiLSTM	0.865	0.834	0.798	0.815	+6.32%
CNN+BiLSTM with more features and data augmentation	0.869	0.827	0.806	0.821	+6.82%
Random Forest Baseline	0.819	0.775	0.723	0.748	
+ NN Embedding	0.841	0.789	0.784	0.787	+2.80%
+ NN Embedding and dist_angle	0.860	0.814	0.808	0.811	+5.03%

6 Conclusion

From our experiments, we find that learning an appropriate representation with respect to the task can be very helpful in classification as the network has more expressive power than pre-trained models. With experiments, we show the embeddings learned by neural networks do significantly improve the accuracy of random forest models. Furthermore, we compare the classification performance of models with and without feature engineering upon those representations, and show that such feature engineering will increase the capability of the models. Based on these observations, we propose two future work directions: 1) to encourage the networks to learn a better task-specific representation, we could employ attention mechanism in our language model so that it could concentrate on the parts of the question that differentiate duplicate question pairs from non-duplicate pairs; 2) to further test if our findings are generalizable to other non-neural network models, we could apply the same learned feature representation and feature engineering strategies to other classifiers, such as SVM and xgboost on the same set of features.

References

Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

Alex Graves, Santiago Fernández, and Jürgen Schmidhuber. Bidirectional lstm networks for improved phoneme classification and recognition. In *International Conference on Artificial Neural Networks*, pages 799–804. Springer, 2005.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

Lili Jiang, Shuo Chang, and Nikhil Dandekar. Semantic question matching with deep learning, Feb 2017. URL `https://engineering.quora.com/Semantic-Question-Matching-with-Deep-Learning`.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014. URL `http://www.aclweb.org/anthology/D14-1162`.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*, 2015.

Abhishek Thakur. is_that_ a_duplicate_quora_question, 2017. URL `https://github.com/abhishekkrthakur/is_that_a_duplicate_quora_question`.