Leveraging Statistic and Semantic Features for Similar Question Detection Using Fusion XGBoost

Siyuan Liao[1], Leung-Pun Wong[2], Lap-Kei Lee[2], Oliver Au[2], and Tianyong Hao[1(✉)]

[1] School of Computer Science, South China Normal University, Guangzhou, China
{sean,haoty}@m.scnu.edu.cn

[2] School of Science and Technology, The Open University of Hong Kong, Hong Kong, China
{s1243151,lklee,oau}@ouhk.edu.hk

Abstract. Question text similarity calculation is a fundamental and essential research problem for community question answering services. Different question text collections have various characteristics. Some frequently answered questions may have distinct statistical patterns, while some questions are syntactically different but semantically similar. To measure question similarity more adaptively to different kinds of question text, this paper proposes a method for identifying similar question utilizing the combination of both statistic and semantic features based on XGBoost. The method extracts semantic and statistical features from question text. After that, a feature set generation method is proposed, along with a model fusion strategy. Based on the standard Yahoo! dataset containing 25,569 questions with answers, three experiments have been conducted to evaluate the performance of the method. Results show that it achieves a precision of 88.65% and a recall of 71.85% outperforming a list of baseline methods.

Keywords: Similar question detection · Feature set generation · XGBoost · Question-answering

1 Introduction

In recent years, Question Answering (QA) has become an important research field of Natural Language Processing (NLP). Detecting similar question is a fundamental problem for Community Question Answering (CQA) systems such as Yahoo! Answers, Baidu Knows and Quora. These systems serve as a platform for users to share their knowledge, allowing users to submit questions or answer questions posted by other users. Due to their popularity, a lot of questions and answers have been recorded and accumulated. The possibility of reusing existing frequently asked questions (FAQ) makes QA systems more convenient and

Y. Nah et al. (Eds.): DASFAA 2020 Workshops, LNCS 12115, pp. 106–120, 2020.
https://doi.org/10.1007/978-3-030-59413-8_9

adaptable. If a question similar to a given new question is identified in the collection of FAQ, corresponding answers can be easily provided by retrieving existing answers associated with the question.

However, the major challenge associated with similar question retrieval is lexico-syntactic gap. Two questions may refer to the same thing but they may differ lexically and syntactically. Thus, many evaluation tasks have been proposed in academia, e.g., SemEval [14], or industries, e.g., Quora[1], where their solutions can become components of automatic methods for detecting duplicate questions.

At present, the main research methods are divided into two categories: traditional machine learning methods and deep learning methods. The performance of traditional methods mainly depend on text representation and similarity metrics, e.g. tf-idf [17] and Word2Vec [13]. Deep learning based models, e.g., [4,16,24], use variants of neural network architectures to model question-question pair similarity. Although deep learning based models have shown their strong ability and convenience in the task, it is hard to achieve the desired accuracy as large training sets are typically not available. Moreover, effectively exploiting full-syntactic parse information in Neural Networks is still an open problem. Traditional methods mainly rely on feature engineering. Based on previous experience, the use of a single similarity metric is prone to incorrect classification of some question pairs.

In this paper, we propose an automated method for similar question detection based on Fusion XGBoost. In order to solve the problem caused by a single feature model, a feature combination method is used to generate candidate feature sets from statistical features and semantic features. We use selected candidate feature sets as input to train the XGBoost model separately, and then adopt a multi-model fusion method based on voting mechanism to enhance the generalization performance of the model. The main contribution of this work lies in two aspects: (1) A statistical and semantic similarity feature combination method is proposed, by which the feature sets generated are more effective. (2) A new model fusion method via XGBoost is proposed, which utilized multiple feature sets.

2 Related Work

Different approaches have been proposed to identify similar questions in CQA. The traditional measures of similarity mainly used metrics based on word frequency calculation and part-of-speech (POS) tagging. Niwattanakul et al. [15] proposed a similarity measurement between keywords and index terms based on Jaccard Coefficient. Huang et al. [8] proposed measurement method of similarity based on tf-idf. Hao et al. [6] proposed an automated approach to detecting similar questions based on the calculation of question topical diversity. Zhou et al. [23] adopted a phrase-based translation model on Yahoo! Answers and evaluated its effectiveness. Wang et al. [21] proposed a model to find semantically

[1] https://www.kaggle.com/c/quora-question-pairs.

related questions by computing similarity between the representing syntactic trees of questions.

There are also a lot of research on measuring semantic similarity of texts. Jiang et al. [9] devised a semantic similarity metric using WordNet. Li et al. [11] devised another semantic similarity metric using HowNet. Li and Zhao [12] used the ontology of domain to calculate semantic similarity. Based on the concept of distributed word vectors, the Word2Vec technique proposed by Mikolov et al. [13] has been sucessfully applied in many NLP tasks. After that, pre-training with large-scale corpora and then tuning on specific tasks has become a mainstream of obtaining semantic representation. Kusner et al. [10] applied word mover's distance to discuss the similarity of two documents. Devlin et al. [5] proposed a new pre-training language representation method and obtain contextual embeddings for sentences, which have been applied to sentence similarity calculation.

Some existing work on detecting similar questions is based on deep learning. Zhou et al. [24] adopt a deep neural network (DNN) based query and answer representation model to rank a set of answers for a given query. Qiu and Huang [16] proposed a convolutional neural tensor network architecture to encode the sentences in semantic space and achieved semantic matching. Das et al. [4] proposed a deep structure topic model to bridge the lexico-syntactic gap between questions. Ruan et al. [18] discussed the methods of calculating the similarity of sentences based on multi-feature fusion. Ye et al. [22] used recurrent neural network (RNN) to measure the semantic similarity between sentences. Chali and Islam [2] applied Long Short-term memory (LSTM) and bi-direction Long Short-term memory (biLSTM) to find the semantic similarity between questions. Uva et al. [20] proposed to inject structural representations in Neural Networks for solving question similarity.

Each method mentioned above has its advantage, but they deal with only a few aspects of sentences. Obviously, questions well matched on different similarity measures are more likely to be similar. Based on this assumption, Song et al. [19] proposed to employ both statistic measure and semantic information. The similarity measure was calculated by combination of dynamically formed vectors and WordNet. Different from the method of simple feature weighting, we propose feature set generation and model fusion methods to utilize multiple different feature sets.

3 Our Approach

The framework of the proposed fusion XGBoost model mainly contains four parts: data preprocessing, feature extraction, feature set generation and model fusion. The overall framework is shown in Fig. 1.

In order to effectively obtain the statistic and semantic features of questions, several data preprocessing strategies are applied. First we remove useless characters other than letters and numbers, unify some common phrases into the same form, and then perform word segmentation and spell correction. After that, statistic and semantic similarity features are extracted from clean question text.

These two types of features are used to initialize the feature sets. The feature set generation algorithm traverses the initial feature sets to generate candidate feature sets. Then, the XGBoost model is trained by selecting feature sets from the candidate feature sets according to the rules of model fusion. After that, a voting fusion based on F1 score is adopted. Finally, the final similar question results are obtained using the fusion model prediction.

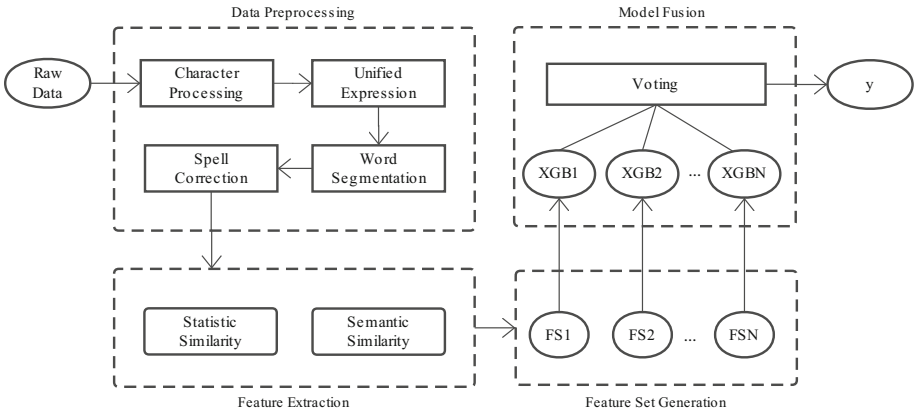

Fig. 1. Overall framework of fusion XGBoost

3.1 Data Preprocessing

For traditional features based on statistical word frequency, the biggest challenge is word sense disambiguation, that is, the same meaning may have different expressions, e.g., different tenses of verbs, singular or plural form of nouns, etc. These differences can be ignored when measuring the similarity of questions. We first remove all characters except letters, numbers and punctuation in the dataset, and then unify all the letters into lowercase.

Unified Expression. Certain phrases have different expressions, such as "*what's*" and "*what is*". The two expressions with the same meaning do not need to be distinguished into different phrases when using word frequency based features. To this end, we adopt a unified expression approach, using regular expressions to unify phrases such as "*what's*", "*can't*" and "*world cup*", replaced with "*what is*", "*cannot*", "*worldcup*". On the other hand, there are Internet catchwords and special domain names in daily life, such as "*4all*", "*any1*", "*6x*", etc. The commonality is that they are all combinations of letters and numbers. We split them and got "*4 all*", "*any 1*", "*6 x*" as the result.

Spell Correction. Text data entered by users on web may have various spelling errors. For example, "*guitars*" is spelled "*guitar*", "*perfect*" is spelled "*perrfect*",

etc. We adopt an open source library pyspellchecker[2] to correct the spelling issues. It applies a Levenshtein Distance algorithm [7] to find permutations within an edit distance of 2 from original word, and then compares all permutations (insertions, deletions, replacements, and transpositions) to known words in a word frequency list. Those words that are found more often in the frequency list are more likely to be the correct results.

3.2 Feature Extraction

The calculation of sentence similarity utilizes a list of similarity metrics. The metrics focusing on either statistic or semantic aspects are used as features.

I. Statistic Similarity Metrics

1) Jaccard Index (Jac)
The Jaccard Index treats a sentence as a collection of words, and then divides the number of intersection elements between two sentences by the number of union elements to get the similarity between the two sentences:

$$Jac(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}, \tag{1}$$

where A, B are respectively the collections of words for the two questions without stop words.

2) Word's Common Rate (CR)
Word's Common Rate is a kind of basic metric to describe the similarity between two collections:

$$CR(A,B) = \frac{|A \cap B|}{|A| + |B|}. \tag{2}$$

3) TF-IDF via Word Match (TWM)
Let $Diff(A)$ denote the word in question A but not in question B, and $Diff(B)$ denote the word in B but not in A. $tfidf(a)$ denote the TF-IDF value of word a. The TWM is calculated using Eq. (3):

$$TWM(A,B) = \frac{\sum_{i=1}^{|C|} tfidf(C_i)}{\sum_{j=1}^{|D|} tfidf(D_i)}, \tag{3}$$

where $C = Diff(A) \cup Diff(B)$ and $D = A \cup B$.

4) Topical Diversity (TD)
Topical words can be generated with different POS tags for a given question. Four combinations of POS tabs are defined as: POS1 (nouns), POS2 (nouns and adjectives), POS3 (nouns and verbs), POS4 (nouns, adjectives and verbs).

[2] https://github.com/barrust/pyspellchecker.

After generating topical words, topical diversity between two questions can be calculated by:

$$TD(A,B) = 1 - \frac{\sum_{t_i \in T_A \cap T_B} tfidf(t_i)}{\sum_{t_j \in T_A \cup T_B} tfidf(t_j)}, \tag{4}$$

where A and B are the two questions, T_A and T_B are the collections of topic words in question A and question B, respectively. From this equation, we can observe that the topical diversity is higher when there are less shared topical words between the questions.

II. Semantic Similarity Metrics

1) Cosine similarity via BERT Sentence Embedding (CosBERT)
BERT pre-training model is adopted to extract the sentence embedding of a given question. The pre-training model is provided by Google Research [5], which gives vector to each word and performs pooling for sentence embedding. The cosine similarity is calculated as:

$$Cosine(V_A, V_B) = \frac{V_A * V_B}{\|V_A\|_2 * \|V_B\|_2} \tag{5}$$

where V_A and V_B are vectors for word A and B, respectively. $||\cdot||$ is Euclidean norm.

2) Cosine similarity via Word2Vec (CosW2V)
The pre-trained word2vec model gives word embedding $V(w)$ for each word in sentence. The sentence embedding for a sentence containing N words w_1, w_2, ..., w_N is calculated as:

$$V = \frac{\sum_{i=1}^{N} V(w_i)}{\sqrt{\sum_{i=1}^{N} V(w_i)^2}} \tag{6}$$

After that, cosine similarity between two questions can be obtained through formula (5).

3) Cosine similarity via Smooth Inverse Frequency weighted Word2Vec (CosSIF)
In order to extract semantic information, the word vector in a sentence is usually averaged as a sentence vector. Averaging gives too much weight to irrelevant words. Arora et al. [1] proposed an algorithm for embedding sentence using word embedding, which uses the smooth inverse frequency feature of the term to weight the word embedding and then average it. After generating all questions' sentence embeddings, PCA/SVD is applied to modify them slightly.

4) Word Mover's Distance (WMD)
The WMD distance measures the dissimilarity between two text documents as the minimum amount of distance that the embedded words of one document need to "travel" to reach the embedded words of another document. The distance between word i and word j becomes $c(i,j) = ||V(i) - V(j)||_2$.

3.3 The Feature Combination Method

Each feature set used in the paper is a combination of similarity metrics. Finding effective feature combinations is the key to improve the performance of the model. Traversing all feature combinations to find the optimal feature combination has a high time complexity. Therefore, a feature set generation algorithm based on pruning strategy is proposed. The detail is shown in Algorithm 1.

Algorithm 1. Feature set generation

Input:
Two feature sets S_1 and S_2;
Output:
Candidate feature sets $Candidates$ and corresponding F1-scores $Fscores$;

$Visited \leftarrow \emptyset$
$Candidates \leftarrow \emptyset$
$Fscores \leftarrow \emptyset$
for each $feature1 \in S_1$ **do**
  for each $feature2 \in S_2$ **do**
    $feature\_set \leftarrow \{feature1, feature2\}$
    $fscore \leftarrow$ **evaluate**$(feature\_set)$
    $Candidates \leftarrow Candidates \cap \{feature\_set\}$
    $Fscores \leftarrow Fscores \cap \{fscore\}$
    $Visited \leftarrow Visited \cap \{feature\_set\}$
  end for
end for
$all\_features \leftarrow S_1 \cap S_2$
for $i = 0$ to $|all\_features| - 2$ **do**
  for each $feature\_set$, $fscore \in Candidates$ and $Fscores$ **do**
    if $|feature\_set| \neq 2 + i$ **then**
      Continue
    end if
    $left\_features \leftarrow all\_features - feature\_set$
    for each $feature \in left\_features$ **do**
      $new\_set \leftarrow feature\_set \cap feature$
      if $new\_set \in Visited$ **then**
        Continue
      end if
      $Visited \leftarrow Visited \cap \{new\_set\}$
      $new\_fscore \leftarrow$ **evaluate**(new_set)
      if $new\_fscore \geq fscore$ **then**
        $Candidates \leftarrow Candidates \cap \{new\_set\}$
        $Fscores \leftarrow Fscores \cap \{new\_fscore\}$
      end if
    end for
  end for
end for

Since containing more features does not ensure the improvement of performance, we propose a method to find the optimal combination of features. The feature set generation algorithm mainly contains two steps: initial feature set

generation and feature set growth. The initial feature set is considered as candidate feature set. Growth is performed on these candidate feature sets based on a total of eight features obtained. If a new feature set obtained achieves better evaluation result, then the new feature set and corresponding evaluation result is added to the candidate feature set. In order to avoid duplication or useless attempts, each new feature set needs to be marked so that the growth of the candidate feature set can be pruned when the algorithm iterates.

The initial feature sets are obtained by cross-combining two types of similarity metrics. After that, their performances on the training set are evaluated, which are shown as lines 1–12. Improved feature sets can be found iteratively based on F-score. The loop statement on line 14 is used to iterate through all possible combinations of features. The statement on line 15 attempts to grow all candidate feature sets. If a new feature set achieves higher F-score, then the feature set is added to the candidate feature sets, otherwise it is discarded and will not be evaluated again, as shown in lines 22–30. Lines 22–25 implement a pruning strategy, ensuring that a feature set without improved F-score will not be tested again.

3.4 XGBoost Model

We used the XGBoost model proposed by Chen [3], which has been widely applied in different kinds of data mining tasks. XGBoost classifier is a boosting classifier which combines hundreds of tree models with lower classification accuracy into a stronger learner in an iterative fashion. At each iteration of gradient boosting, the residual is used to correct previous predictor such that the specified loss function can be optimized. Different from other Gradient Boosting Decision Tree (GBDT) model, regularization is added to the loss function to establish the objective function in XGBoost, which is given by:

$$\begin{aligned} J(\Theta) &= L(\Theta) + \Omega(\Theta), \\ L(\Theta) &= \sum_{i=1}^{n} l(\widehat{y_i}, y_i), \\ \Omega(\Theta) &= \sum_{k=1}^{t} \tau T + \frac{1}{2}\lambda ||\boldsymbol{w}||^2 \end{aligned} \tag{7}$$

Here, Θ refers to the various parameters in the formula. $J(\Theta)$ is the objective function. $L(\Theta)$ is the training loss function that measures the difference between the prediction $\widehat{y_i}$ and the target y_i. Commonly used convex loss function such as square loss or logistic loss can be used in the above equation. $\Omega(\Theta)$ is a regularized term that penalizes complex models. In the definition of $\Omega(\Theta)$, T is the number of leaves in the tree, τ is the learning rate, λ is a regularized parameter to scale the penalty, and $\boldsymbol{w}$ is the weight of the leaves. Since the base model is decision tree, the result of prediction $\widehat{y_i}$ is the sum of scores predicted by K trees:

$$\widehat{y_i} = \sum_{k=1}^{t} f_k(x_i), f_k \in F \tag{8}$$

where x_i is the i-th training sample, F is the space of functions containing all regression trees and $f_k(x_i)$ is the score for the k-th tree. The optimization goal is to construct a tree structure that minimizes the target function in each iteration. The tree structure learns from conclusions and residuals of previous trees, fitting a current residual regression tree.

3.5 The Multi-model Fusion Method

In order to acquire better performance, a multi-model fusion method is proposed. Model fusion is an effective way to improve the accuracy of model, and voting weighted fusion is a fast and direct method. After feature sets are generated, the models obtained by different feature sets can be used for model fusion. The basic element of model fusion is to ensure that the correlation between individual models is small, and the performance difference between different models is not significant. Therefore, models obtained by different feature sets are used for model fusion.

The candidate feature sets are obtained through the feature set generation algorithm. After that, the first step is to generate multiple XGBoost models with different feature sets as input, and then calculate the value of F1 for each model respectively. Let s_i represents a collection of feature sets consisting of i features, p_i is the selected feature set. Then

$$p_i = \arg\max_{j \in s_i} \ fscore(j) \tag{9}$$

The second step is the selection of the models. Each of these models is an XGBoost model trained by an optimal feature set of a specific number of features. The model fusion method is to conduct weighted voting according to the F1 score of the selected model. Assume that N' models are selected, the final prediction of sample i is:

$$\widehat{y_i} = \sum_{j=1}^{N'} \widehat{y_i}(j) w_j \tag{10}$$

In this formula, $\widehat{y_i}(j)$ is prediction for sample i given by model j, w_j is the voting weight of the model j. Models with higher F1 scores usually have better classification performance, so w is calculated based on the F1 score of the model on the training set. For $j \in [1, N']$:

$$w_j = \frac{fscore(j)}{\sum_{i=1}^{N'} fscore(i)} \tag{11}$$

4 Evaluation and Results

4.1 Datasets

We applied the same QA dataset as used in existing baseline methods, e.g., [6]. The dataset was generated from 25569 questions groups that shared the same

answer extracted in Yahoo! Answers. We filtered the answers that were too short, e.g, "*Yeah*", and obtained a total of 624 question groups. Eventually, the dataset contained questions with a maximum length of 25 words, a minimum length of 2 words, an average length of 10 words, and a standard deviation of 4.50.

We divided the dataset into a training set containing 524 question groups and a testing set containing 100 question groups randomly, similar to the baseline method. Since the number of samples in the dataset was relatively small, the training results fluctuated greatly by different division methods. The experiment were randomly repeated 100 times and the average of the evaluation results was considered.

4.2 Baselines

The baseline methods for comparison were widely used text similarity calculation measures as follows:

1) Topical Diversity: We used TD+POS4 method described in Sect. 3.2.
2) Jaccard: Described in Sect. 3.2.
3) CosBERT: Single question similarity detection model using feature described in Sect. 3.2.
4) WMD: Unsupervised method to calculated the semantic distance between two questions using Word2Vec model. Described in Sect. 3.2.
5) CosSIF: Described in Sect. 3.2.
6) CosW2V: Described in Sect. 3.2.
7) Metric Longest Common Subsequence: The ratio of the number of words in the longest subsequence of two questions to the number of words in the longest question.
8) NN: A multi-feature fusion method based on neural network proposed by Ruan et al. [18].

Feature extraction involves two pre-training models with training parameters and sources as follows. Word2Vec [13] was trained on Google News dataset, which contains 300-dimensional vectors for 3 million words and phrases. The resulting vectors have been made publicly available[3]. BERT [5] was multilingual uncased BERT-Base model[4]. The model gives 768-dimensional vectors for each word.

In the experiments, we used the same model configuration for model training of each feature set. The learning rate and was set to 0.05. The maximum depth of tree in each booster and was set to 3. Evaluation metric was a logarithmic loss function. These parameters were set empirically.

4.3 Results

Four experiments were conducted to evaluate the effectiveness of the proposed method for similar question detection. The evaluation metrics were precision, recall, and F1 score, which were commonly used in information retrieval area.

[3] see https://code.google.com/archive/p/word2vec/.
[4] see https://github.com/google-research/bert.

Table 1. The performance comparison with the different feature sets (%)

Features	Methods	Avg precision	Avg recall	Avg F1
TWM, Jac, CosSIF	Random Forest	84.52	70.98	76.43
CosBERT, TD, Jac, CosSIF		83.61	71.31	76.32
Jac, CosSIF		82.89	71.88	76.34
TWM, Jac, CosSIF	SVM	93.20	53.63	67.42
CosBERT, TD, Jac, CosSIF		**94.28**	53.50	67.56
Jac, CosSIF		91.94	55.59	68.67
TWM, Jac, CosSIF	XGBoost	85.37	72.64	**77.99**
CosBERT, TD, Jac, CosSIF		85.30	**72.74**	77.97
Jac, CosSIF		83.34	72.22	76.77

First, an experiment to evaluate the performance of feature combinations was conducted. According to the feature set generation algorithm, candidate feature sets sorted according to F1 score can be obtained. Then the feature sets with the highest F1 score of two, three and four features were selected, namely FS1(Jac + CosSIF), FS2(TWM + Jac + CosSIF) and FS3(CosBERT + TD + Jac + CosSIF). These three feature sets were then used to train different classifiers, including SVM, Random Forest, and XGBoost, and compare their performances. The result showed that the feature set FS2 achieved the highest F1 in the XGBoost classifier. The feature set with the largest number of features in candidate sets has five features, but the F1 score is lower than that of the feature set with only two features in SVM and Random Forest. The result is shown as Table 1. From the result, the XGBoost model with TWM, Jac and CosSIF as feature set achieves the highest average F1.

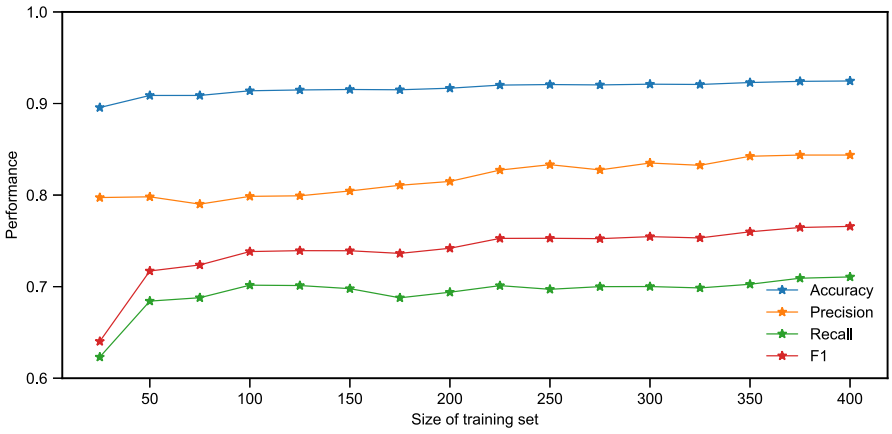

Fig. 2. The performance of fusion XGBoost training using different sizes of data

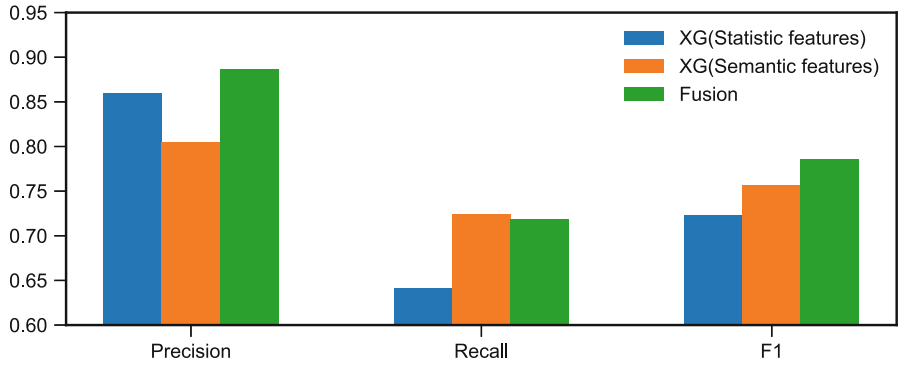

Fig. 3. The performance in datasets using different types of features

The second experiment evaluates how question similarity calculation performance was affected by the size of training datasets to test the scalability of our model. The used datasets were the same as used in the first experiment.

As shown in Fig. 2, when the number of samples in the training set was more than 225, the performance of the model became stable. When the number of samples was less than 225, the performance of the model fluctuated significantly. This demonstrated that the model was effective when the number of training set samples was more than 225.

We compared the performance obtained by single XGBoost model using two different types of features as input. The XG (Statistic features) model using all features described in our statistic similarity metrics, and XG (Semantic features) using all semantic similarity metrics. The Fusion model using our proposed fea-

Table 2. The performance comparison with baselines (%)

Methods	Avg accuracy	Avg precision	Avg recall	Avg F1
CosSIF	87.17	60.72	70.91	64.95
Metric Longest Common Subsequence	89.64	70.45	67.58	68.49
WMD	90.41	74.89	70.57	70.89
CosW2V	90.65	75.02	69.17	71.02
CosBERT	90.37	71.79	72.13	71.49
Jaccard	91.05	77.39	67.68	71.63
Topical Diversity (TD)	91.33	82.07	65.10	71.49
NN	62.15	48.75	**76.68**	51.40
XG (TWM+Jaccard+CosSIF)	93.17	85.37	72.64	77.99
XG (CosBERT+TD+Jaccard+CosSIF)	93.14	85.30	72.74	77.97
XG (Jaccard+CosSIF)	92.74	83.34	72.22	76.77
Fusion (FS1+FS2)	92.80	**89.56**	69.49	77.42
Fusion (FS1+FS2+FS3)	**94.60**	88.65	71.85	**78.54**

ture combination and multi-model fusion method. According to Fig. 3, the Fusion model achieved highest F1, which demonstrated the effectiveness of our proposed feature combination and multi-model fusion method.

We then compared our method with baseline methods. The result was shown in Table 2. Compared with the TD+POS4 algorithm, our approach of XGBoost single model using the feature group TWM+Jaccard+CosSIF achieves 4.02% precision improvement, 11.58% recall improvement and 9.09% F1 improvement. The fusion model of the three different length feature sets reached the highest precision and F1, and F1 is increased by 9.86% compared with TD+POS4.

4.4 Error Analysis

The result errors including undetected and incorrectly identified similar questions in the experiments were collected and systematically analyzed. Since the models were evaluated by randomly dividing the data set 100 times, we calculated the cases of prediction errors in 100 tests using the fusion model FS1 + FS2 + FS3.

16 groups of questions were predicted to be incorrect more than 10 times, and a total of 5 groups of questions were detected by our method. We found it is difficult for our model to detect similar question pairs with uncommon words, non-generic words, and abbreviations, e.g., PSP and SUV. Two samples of this type of error are shown in Table 3.

Table 3. Examples of undetected error by our method

Group ID	Questions	Answer
G1	*Where can i find free downloads for my PSP?* *I saw an advert for a psp related site for downloads it had 2 dust balls or something like that whats the site*	*www.yourpsp.com*
G2	*What is SUV?* *SUV meaning?*	*Sports utilize vehicle*

11 groups of questions that were incorrectly judged to be similar by our model. There was a common denominator of this type of error, where one question was much longer than the other and the longer one contained the same or related words of the short questions. For instance, the question "*Does anyone know this logo?*" and the question "*anybody know what the company logo is with the singing parrot?*" were marked as similar questions but were annotated dissimilar by human annotators. By analyzing this type of errors, we found that some question pairs had a high proportion of identical or related words, resulting in close semantic features. Therefore, the question target or answer type may be a key factor for eliminating the effect of the word overlap.

5 Conclusion

In this paper, we have applied a fusion XGBoost model to detect similar questions. A feature set generation and a voting method based on F1 score of each model are proposed. Experiments on Yahoo! dataset have showed that the proposed method is feasible in improving the performance of similar question detection. In the future, we plan to evaluate the proposed method on more datasets and consider more features of questions.

Acknowledgement. This work was supported by National Natural Science Foundation of China (No.61772146), Natural Science Foundation of Guangdong Province (2018A030310051), and the Katie Shu Sui Pui Charitable Trust – Research and Publication Fund (KS 2018/2.8).

References

1. Arora, S., Liang, Y., Ma, T.: A simple but tough-to-beat baseline for sentence embeddings (2016)
2. Chali, Y., Islam, R.: Question-question similarity in online forums. In: The 10th Annual Meeting of the Forum for Information Retrieval Evaluation, pp. 21–28. ACM (2018)
3. Chen, T., Guestrin, C.: XGBoost: a scalable tree boosting system. In: The 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 785–794. ACM (2016)
4. Das, A., Shrivastava, M., Chinnakotla, M.: Mirror on the wall: finding similar questions with deep structured topic modeling. In: Bailey, J., Khan, L., Washio, T., Dobbie, G., Huang, J., Wang, R. (eds.) PAKDD 2016. LNCS, vol. 9652, pp. 454–465. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-31750-2_36
5. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: pre-training of deep bidirectional transformers for language understanding. arXiv:1810.04805 (2018)
6. Hao, T., Li, C., Liang, W., Qu, Y.: A topical diversity-based approach to detecting similar question groups from collaborative question-answering archives. In: Web Intelligence, vol. 14, pp. 301–308. IOS Press (2016)
7. Heeringa, W.J.: Measuring dialect pronunciation differences using Levenshtein distance. Ph.D. thesis. Citeseer (2004)
8. Huang, C.H., Yin, J., Hou, F.: A text similarity measurement combining word semantic information with TF-IDF method. Chin. J. Comput. **34**(5), 856–864 (2011)
9. Jiang, J.J., Conrath, D.W.: Semantic similarity based on corpus statistics and lexical taxonomy. arXiv preprint cmp-lg/9709008 (1997)
10. Kusner, M., Sun, Y., Kolkin, N., Weinberger, K.: From word embeddings to document distances. In: ICML, pp. 957–966 (2015)
11. Li, S., Zhang, J., Huang, X., Bai, S., Liu, Q.: Semantic computation in a Chinese question-answering system. J. Comput. Sci. Technol. **17**(6), 933–939 (2002)
12. Li, W., Zhao, Y.: Semantic similarity between concepts algorithm based on ontology structure. Jisuanji Gongcheng/ Comput. Eng. **36**(23) (2010)
13. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in Neural Information Processing Systems, pp. 3111–3119 (2013)

14. Nakov, P., et al.: SemEval-2017 task 3: community question answering. arXiv preprint arXiv:1912.00730 (2019)
15. Niwattanakul, S., Singthongchai, J., Naenudorn, E., Wanapu, S.: Using of Jaccard coefficient for keywords similarity. In: The International Multiconference of Engineers and Computer Scientists, vol. 1, pp. 380–384 (2013)
16. Qiu, X., Huang, X.: Convolutional neural tensor network architecture for community-based question answering. In: IJCAI (2015)
17. Ramos, J., et al.: Using TF-IDF to determine word relevance in document queries. In: The First Instructional Conference on Machine Learning, Piscataway, NJ, vol. 242, pp. 133–142 (2003)
18. Ruan, H., Li, Y., Wang, Q., Liu, Y.: A research on sentence similarity for question answering system based on multi-feature fusion. In: 2016 IEEE/WIC/ACM International Conference on Web Intelligence (WI), pp. 507–510. IEEE (2016)
19. Song, W., Feng, M., Gu, N., Wenyin, L.: Question similarity calculation for FAQ answering. In: SKG, pp. 298–301. IEEE (2007)
20. Uva, A., Bonadiman, D., Moschitti, A.: Injecting relational structural representation in neural networks for question similarity. In: ACL, pp. 285–291 (2018)
21. Wang, K., Ming, Z., Chua, T.S.: A syntactic tree matching approach to finding similar questions in community-based QA services. In: The 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 187–194. ACM (2009)
22. Ye, B., Feng, G., Cui, A., Li, M.: Learning question similarity with recurrent neural networks. In: 2017 IEEE ICBK, pp. 111–118. IEEE (2017)
23. Zhou, G., Cai, L., Zhao, J., Liu, K.: Phrase-based translation model for question retrieval in community question answer archives. In: ACL, pp. 653–662 (2011)
24. Zhou, G., Zhou, Y., He, T., Wu, W.: Learning semantic representation with neural networks for community question answering retrieval. Knowl.-Based Syst. **93**, 75–83 (2016)