IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received January 8, 2020, accepted January 16, 2020, date of publication January 21, 2020, date of current version February 11, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2968391

Duplicate Question Detection With Deep Learning in Stack Overflow

LITING WANG, LI ZHANG, AND JING JIANG
State Key Laboratory of Software Development Environment, Beihang University, Beijing 100191, China

Corresponding author: Jing Jiang (jiangjing@buaa.edu.cn)

This work was supported in part by the National Key Research and Development Program of China under Grant 2018AAA0102304, in part by the National Natural Science Foundation of China under Grant 61672078, and in part by the State Key Laboratory of Software Development Environment under Grant SKLSDE-2019ZX-05.

ABSTRACT Stack Overflow is a popular Community-based Question Answer (CQA) website focused on software programming and has attracted more and more users in recent years. However, duplicate questions frequently appear in Stack Overflow and they are manually marked by the users with high reputation. Automatic duplicate question detection alleviates labor and effort for users with high reputation. Although existing approaches extract textual features to automatically detect duplicate questions, these approaches are limited since semantic information could be lost. To tackle this problem, we explore the use of powerful deep learning techniques, including Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN) and Long Short-Term Memory (LSTM), to detect duplicate questions in Stack Overflow. In addition, we use Word2Vec to obtain the vector representations of words. They can fully capture semantic information at document-level and word-level respectively. Therefore, we construct three deep learning approaches WV-CNN, WV-RNN and WV-LSTM, which are based on Word2Vec, CNN, RNN and LSTM, to detect duplicate questions in Stack Overflow. Evaluation results show that WV-CNN and WV-LSTM have made significant improvements over four baseline approaches (i.e., DupPredictor, Dupe, DupPredictorRep-T, and DupeRep) and three deep learning approaches (i.e., DQ-CNN, DQ-RNN, and DQ-LSTM) in terms of recall-rate@5, recall-rate@10 and recall-rate@20. Furthermore, the experimental results indicate that our approaches WV-CNN, WV-RNN, and WV-LSTM outperform four machine learning approaches based on Support Vector Machine, Logic Regression, Random Forest and eXtreme Gradient Boosting in terms of recall-rate@5, recall-rate@10 and recall-rate@20.

INDEX TERMS Stack overflow, duplicate question detection, deep learning.

I. INTRODUCTION

There are some Community-based Question Answering (CQA) websites that are becoming increasingly popular, such as Quora,[1] Yahoo! Answers,[2] and Stack Overflow.[3] Stack Overflow is a CQA website about software programming. Each user can freely post questions in Stack Overflow. As of October 2019, Stack Overflow had more than 18 million questions. Although the posting ethics were guided in detail, the quality of many posted questions is poor [1]. Even if the users are reminded to search for a forum before posting a new question, there are numerous duplicate questions which were previously posted and answered in Stack Overflow. To reduce the number of duplicate questions, users with high reputation in Stack Overflow manually mark the duplicate questions, resulting in users spending a lot of time and effort. What's more, a large number of duplicate questions are still not detected for a long time. Ahasanuzzaman *et al.* reported that more than 65% of duplicate questions took at least one day to be closed, and a large part of duplicate questions are closed after one year [2]. Therefore, an automatic duplicate question detection approach is required to automatically detect duplicate questions in Stack Overflow.

Some automatic duplicate question detection approaches on Stack Overflow website have been studied in previous works. Zhang *et al.* proposed a DupPredictor approach to automatically detect duplicate questions in Stack Overflow,

The associate editor coordinating the review of this manuscript and approving it for publication was Muhammad Asif.

[1]https://www.quora.com
[2]https://answers.yahoo.com
[3]https://stackoverflow.com

which took into account the similarity features of topics, titles, descriptions, and tags of each question pair [3]. Ahasanuzzaman *et al.* proposed a Dupe approach by considering five features to detect duplicate questions in Stack Overflow [2]. These five features contain cosine similarity value, term overlap, entity overlap, entity type overlap, and wordNet similarity features respectively. Silva *et al.* built two reproduction approaches DupPredictorRep-T and DupeRep based on DupPredictor [3] and Dupe [2] to detect duplicate questions in Stack Overflow [4]. Although these existing approaches have solved the problem of automatically detecting duplicate questions in Stack Overflow, they are limited since semantic information could be lost.

Currently, traditional machine learning techniques and deep learning techniques have been widely applied in the natural language processing tasks, such as text classification [5] and sentiment analysis [6]. The traditional machine learning approaches sometimes performed better than deep learning approaches. However, deep learning has also been used to solve some software engineering tasks, such as code clone detection [7], bug reports detection [8], predicting semantically linkable knowledge [9] and software defect prediction [10]. They have been proven to be effective for some software engineering tasks [11]. In our previous work [12], we use traditional machine learning techniques and deep learning techniques to detect duplicate questions in Stack Overflow. We found that deep learning approaches are more effective than traditional machine learning approaches in the duplicate question detection tasks and they can fully capture the document-level semantics.

Furthermore, Word2Vec is widely used to obtain the vector representations of words in text classification and it can fully capture the semantic information at word-level [13], [14]. Therefore, we use deep learning approaches and Word2Vec to solve the problem of duplicate question detection in Stack Overflow. They can fully capture the semantic information at document-level and word-level respectively. In this paper, we construct three different deep learning approaches based on three different deep learning models and Word2Vec to detect duplicate questions in Stack Overflow. Three different deep learning models are Convolutional Neural Networks (CNN) [15], Recurrent Neural Networks (RNN) [16], and Long Short-Term Memory (LSTM) [17]. Our three approaches are a WV-CNN approach based on Word2Vec and CNN, a WV-RNN approach based on Word2Vec and RNN, and a WV-LSTM approach based on Word2Vec and LSTM, respectively. They are used to predict whether a pair of questions is duplicate (positive) and nonduplicate (negative).

In order to evaluate the effectiveness of our approaches (i.e., WV-CNN, WV-RNN, and WV-LSTM), 134,246 non-master questions and 88,476 master questions in Stack Overflow are collected. Then, the questions of six different question groups that are tagged with Java, Html, Python, C++, Ruby and Objective-C are extracted from these questions as our experimental datasets.

The main contributions of this paper are following (the new contributions that extend our previous works [12] are highlighted in bold font):

- In our previous work [12], three deep learning approaches (i.e., DQ-CNN, DQ-RNN, and DQ-LSTM) are explored based on CNN, RNN and LSTM to solve the problem of duplicate question detection in Stack Overflow. **To further capture the semantic information at word-level, Word2Vec is used to learn word embedding. We construct three approaches WV-CNN, WV-RNN and WV-LSTM based on Word2Vec, CNN, RNN and LSTM to detect duplicate questions in Stack Overflow.**
- Four different machine learning approaches (i.e., Support Vector Machine (SVM) [18], Logic Regression (LR) [19], Random Forest (RF) [20], and eXtreme Gradient Boosting (Xgboost) [21]) are used to detect duplicate questions in our previous work [12]. **We compare these four machine learning (i.e., SVM, LR, RF, and Xgboost) with our three approaches (i.e., WV-CNN, WV-RNN, and WV-LSTM) to analyze the effectiveness of traditional machine learning approaches and deep learning approaches in duplicate question detection tasks.**
- **Our three approaches WV-CNN, WV-RNN and WV-LSTM are also compared with four baseline approaches (i.e., DupPredictor [3], Dupe [2], DupPredictorRep-T [4], and DupeRep [4]) and three deep learning approaches (i.e., DQ-CNN [12], DQ-RNN [12], and DQ-LSTM [12]), respectively. Results show that our approaches have good improvements on duplicate question detection in Stack Overflow.**

The rest of this paper is organized as follows. Section II introduces background knowledge. Section III details our three approaches used to detect duplicate questions in Stack Overflow. We report the evaluation results in Section IV. The threats to validity are discussed in Section V, and related works are introduced in Section VI. Finally, Section VII concludes this paper.

II. BACKGROUND

In this section, we describe four necessary background knowledge. There are duplicate questions, deep learning, word embedding, and data collection respectively.

A. DUPLICATE QUESTIONS

Stack Overflow is one of the most popular CQA based on the software programming. Stack Overflow has more than 18 million questions, 28 million answers, 76 million comments and 56,000 tags as of October 2019. In Stack Overflow, each question contains many attributes, such as *ID*, *title*, *body*, *tags*, *creationDate*, *closedDate*, etc. Although users are reminded to search for the forum before creating a new question, duplicate questions frequently appear in Stack Overflow. Duplicate questions refer to questions that were previously created and answered in Stack Overflow. In order to reduce the number of duplicate questions, the users with high reputation are encouraged to manually

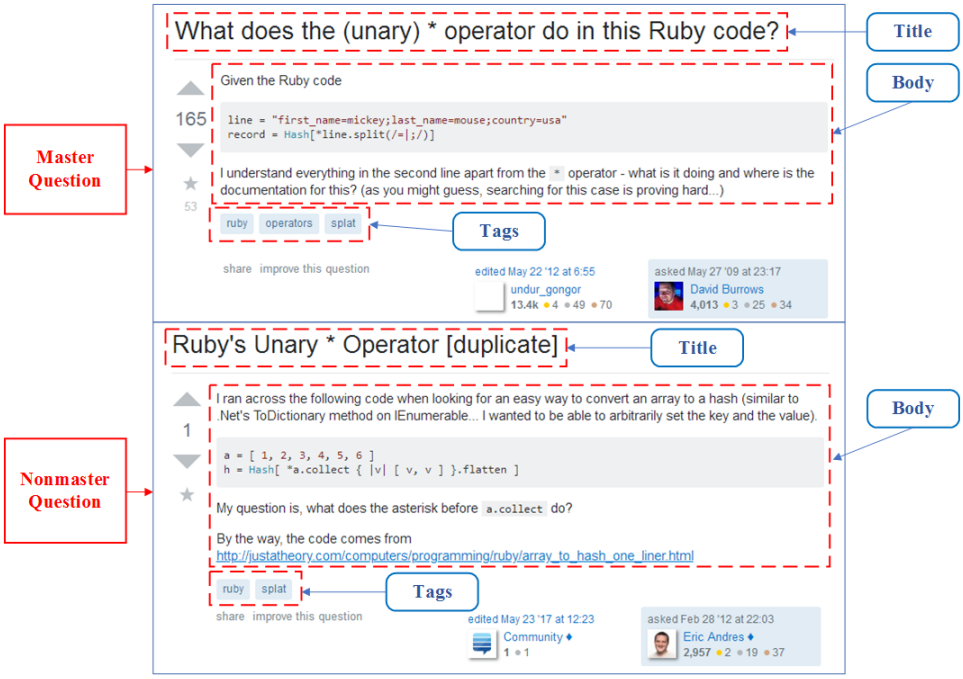

FIGURE 1. An example of a duplicate question pair.

mark duplicate questions in Stack Overflow. Generally, if two questions are duplicates, one of them will be marked as a nonmaster question (duplicate question) and closed. The other question will be marked as a master question. Moreover, a pair of duplicate questions consists of a master question and a nonmaster question. In a pair of duplicate questions, the older question that is created is called the master question, and the recent question that is created is considered as the nonmaster question. Fig.1 is an example of a duplicate question pair manually marked by a user with high reputation. The duplicate question pair includes a master question with ID 918449 and a nonmaster question with ID 9490456. We can see that the title, body, and tags of a nonmaster question are identical to its master question. These two questions are all about solving the problem of ''Ruby's Unary * Operator''. The user with high reputation in Stack Overflow manually marks a nonmaster question as a duplicate question of the corresponding master question [2]. The title of each duplicate question is marked with ''[duplicate]'', and all duplicate questions are closed by users with high reputation. Therefore, we extract the duplicate questions based on the ''[duplicate]'' marked in title of each question.

B. DEEP LEARNING

The deep learning is an increasingly popular technique and has been widely applied in different domain tasks, such as natural language processing tasks and software engineering tasks. Various deep learning models have been proposed, and they can identify hidden patterns, underlying dynamics and semantic information of data by self-learning processing. For example, there are three popular deep learning models that are CNN [15], RNN [16], and LSTM [17] respectively. They are widely used to solve the natural language processing tasks such as text classification [5], and sentiment analysis [6], etc. They are also used to solve some software engineering tasks, such as code clone detection [7], bug reports detection [8], predict semantically linkable knowledge [9], and software defect prediction [10]. CNN is a feed-forward multilayer neural network consisting of one or more convolutional and subsampled layers with fully connected layers optionally. CNN has significantly improved over traditional natural language processing techniques in the task of sentence classification and sentiment analysis [8]. RNN is an effective architecture for sequence learning tasks in which the data is highly correlated along a single axis. RNN has been successfully used for various tasks, such as language modeling and learning word embedding [9]. Moreover, LSTM can solve a large number of tasks that were not solved by previous learning algorithms for RNN [10]. As more and more researchers believe that deep learning is superior to the traditional techniques for solving software engineering problems [22]. Therefore, we explore three deep learning approaches based on above three popular deep learning models to solve the problem of duplicate question detection in Stack Overflow.

C. WORD EMBEDDING

Word Embedding[4] is the collective name for a set of language modeling and feature learning techniques in natural language processing where words or phrases from the vocabulary are mapped to vectors of real numbers. Some previous works on word embedding includes Neural Network Language

[4]https://en.wikipedia.org/wiki/Word_embedding

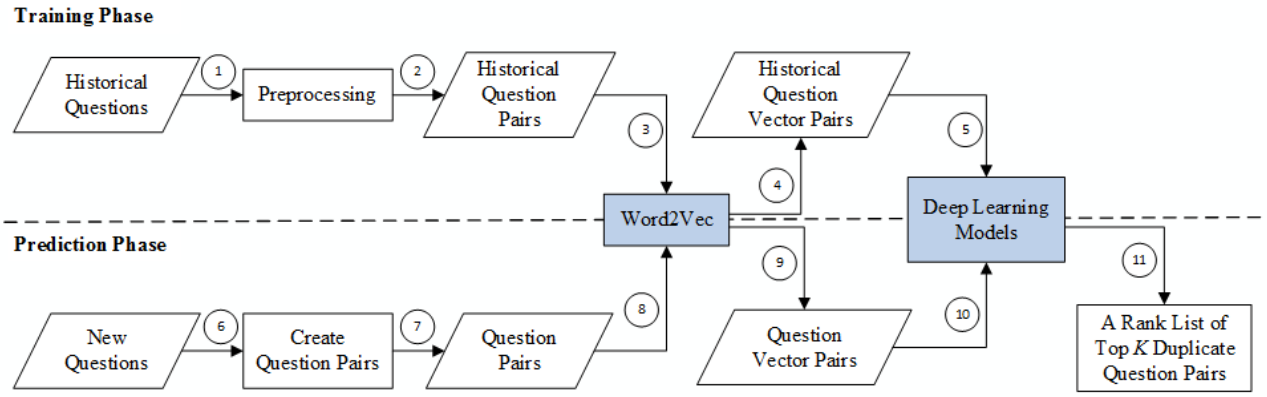

FIGURE 2. The overall framework of our approaches.

Model (NNLM) [23], [24], Latent Semantic Analysis (LSA) [25], and Latent Dirichlet Allocation (LDA) [26], etc. Recently, Word2Vec is widely used to learn word embedding and can translate a word into a numerical vector [13]. In particular, Word2Vec can fully capture the word-level semantics. Word2vec includes Continuous Bag-of-Words (CBOW) architecture and Skip-gram architecture. CBOW and Skip-gram architectures are used to calculate the vector representations of words, which not only improve the quality of word vectors but also reduce the computational complexity.[5] Thus, in our work, we use Word2Vec to obtain the vector representations of words, and CBOW architecture is used to learn high-quality word vectors.

D. DATA COLLECTION

The data of Stack Overflow are publicly available datasets provided by Creative Commons Data Dump Service.[6] At first, all questions created from August 2008 to September 2014 in Stack Overflow are collected. Secondly, we extract all nonmaster questions that are closed as duplicate questions and appended "[duplicate]" to their titles from these questions. Thirdly, in order to obtain their master questions that these nonmaster questions are linked, we check the *LinkTypeId* of the *postslinks* table to identify them. Finally, we collect 134,261 nonmaster questions and 88,476 master questions. As previous work [4], we extract the questions of six different question groups tagged with Java, Html, Python, C++, Ruby, and Objective-C respectively. The textual content of each question contains the title, body, and tags. Moreover, according to previous work [4], the question pairs based on the datasets of these six different question groups are created as our experimental datasets. The question pairs include duplicate question pairs (positive examples) and nonduplicate question pairs (negative examples), respectively. To avoid bias, the number of nonduplicate question pairs is the same as the number of duplicate question pairs. For six different question groups, we use 80% of duplicate question pairs and nonduplicate question pairs as our training data, and the remaining 20% of duplicate question pairs are used as testing data.

[5]https://code.google.com/p/word2vec/
[6]http://archive.org/details/stackexchange

III. APPROACH

In this section, three deep learning approaches WV-CNN, WV-RNN and WV-LSTM based on Word2Vec, CNN, RNN and LSTM are constructed to solve the problem of duplicate question detection in Stack Overflow. At first, we introduce the overall framework of our approaches in detail. Then, we describe the problem formation and word embedding. Finally, WV-CNN, WV-RNN, and WV-LSTM are elaborated respectively.

A. OVERALL FRAMEWORK

As shown in Fig. 2, the overall framework of our approaches consists of two phases, which are model training phase and model prediction phase respectively.

In the model training phase, our framework first collects historical questions from Stack Overflow. Then, the questions are preprocessed (Step 1). In the preprocessing step, we first extract the title, body and tags of each question. Next, we remove stop words and preform stemming for the title and body of each question. After we have preprocessed the questions, historical question pairs are created (Step 2). These question pairs consist of duplicate question pairs and nonduplicate question pairs. We use Word2Vec to obtain vector representations of words (Step 3), and then the word vector representations of all historical question pairs are acquired (Step 4). Finally, the vectors of all question pairs are fed into the deep learning models to train models (Step 5).

In the prediction phase, each new question pair is classified by the trained deep learning model and a probability distribution is obtained. At first, We create test question pairs by pairing each new question with each previous question (Step 6). Then, we obtain the question pairs that are used for model prediction (Step 7). The question pairs are trained by using Word2Vec (Step 8). Next, the word vector representations of all question pairs are acquired (Step 9). In addition,

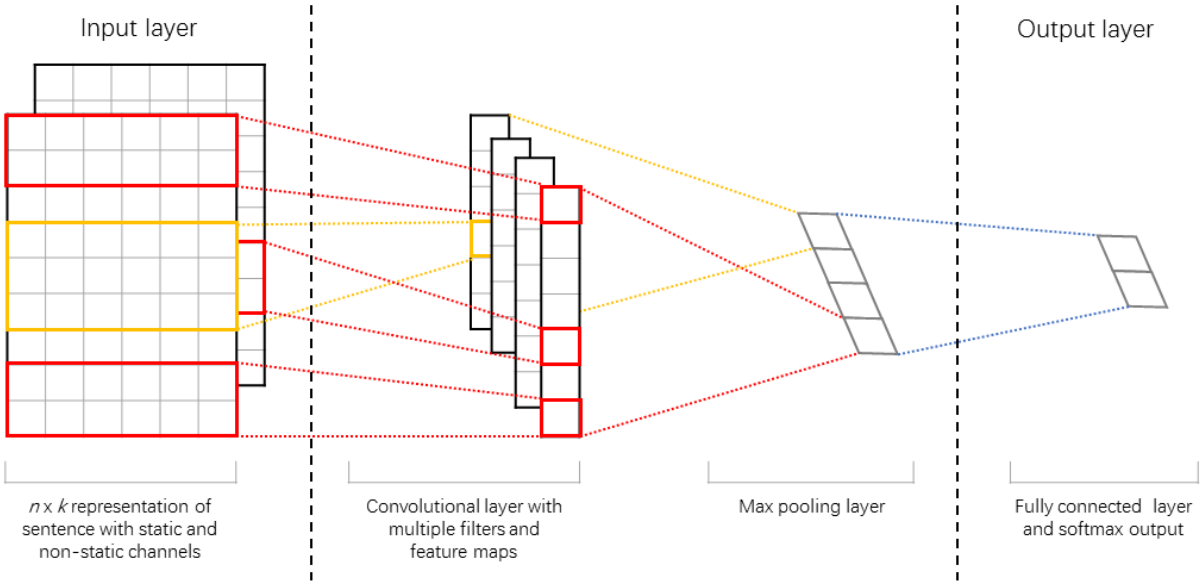

FIGURE 3. The architecture of a convolutional neural network [5].

the question vector pairs are fed into the trained deep learning models to classify and obtain the probability distribution of all duplicate question pairs (Step 10). Finally, a rank list of top K duplicate question pairs is obtained by using highest probability ranking (Step 11).

B. PROBLEM FORMULATION AND WORD EMBEDDING

Given a question q_i, the attributes of q_i contains an identifier id, a body b, a title t, a set of tags T, etc. We suppose that the title t, body b and a set of tags T are used as the description d_i of q_i. Similarly, d_j denotes the description of a question q_j. We then build a question pair r_{ij} by combining q_i and q_j.

Word embedding is utilized to convert tokenized words into real-valued feature vectors [27]. Word2Vec is widely used to learn word embedding and it can fully capture the word-level semantics. To fully capture the word-level semantics, we use Word2Vec to obtain word vector representations of each question pair. The word vector representations of each question pair will be used as input of the deep learning models. We assume that each question pair consists the content of titles, bodies, and tags. For a given question pair r_{ij}, the text sequence is respected as $\{x_1, x_2, \cdots, x_m\}$. m is the length of r_{ij}. Let $x_i \in \mathbb{R}^n$ be the n-dimensional word vector corresponding to the ith word in r_{ij}. The word vector of r_{ij} is represented as follows:

$$x_{1:m} = x_1 \oplus x_2 \oplus \cdots \oplus x_m, \tag{1}$$

where $\oplus$ is the concatenation operator and $x_{i:i+j}$ refers to the concatenation of words $x_i, x_j, \cdots, x_{i+j}$. It can be represented by a $m \times n$ matrix.

C. WV-CNN

Convolutional Neural Networks (CNN) [15] are proven to be effective for the natural language processing. We construct a WV-CNN approach based on Word2Vec and CNN to detect duplicate questions in Stack Overflow.

In text classification, each question pair is represented as a matrix. The input layer consists of a matrix, and each row in the matrix is the vector representation of a word. In the convolution layer, multiple filters slide over the rows of matrices. Then, the max value in each filter is extracted by the max pooling layer. The output layer is a softmax classifier. Fig. 3 illustrates all the steps of a Convolutional Neural Network.

Given a question pair r_{ij}, let $x_{1:m}$ be the word vector representations of r_{ij}. A convolution filter w uses a region of k words to produce a new feature. For instance, a feature f_i is generated from a region of words $x_{i:i+k-1}$, and it is defined as follows:

$$f_i = tanh(w \cdot x_{[i:i+k-1]} + b), \tag{2}$$

where $tanh$ is a non-linear hyperbolic tangent function, and $b \in \mathbb{R}$ is a bias term.

As one filter can produce one feature, multiple features are obtained by multiple filters with different region sizes. These filters are applied to each possible region of words to produce a feature map. A feature map c is defined as follows:

$$c = [c_1, c_2, \ldots, c_{m-k+1}], \tag{3}$$

where $c \in \mathbb{R}^{m-k+1}$. A 1-max pooling operation is applied to the feature map and the maximum value c' is obtained by $c' = max\{c\}$ as the feature. Then, the multiple features z are obtained by each feature, namely $z = |c'_1, c'_2, \cdots, c'_e|$.

The probability distribution P_c of a given question pair r_{ij} is calculated by the softmax function. P_c is defined as follows:

$$P_c = softmax(U \cdot z + B), \tag{4}$$

where U is a weight vector, and B is a bias term.

A rank list of top K duplicate question pairs is obtained by highest probability ranking and is used to detect whether there is a duplicate question pair consisting of a given question and its master question.

D. WV-RNN

Recurrent Neural Networks (RNN) [16] are suited for processing sequential text data in natural language processing tasks. We construct a WV-RNN approach based on Word2Vec and RNN for solving the problem of duplicate question detection in Stack Overflow.

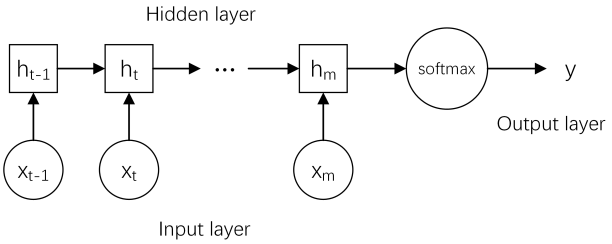

FIGURE 4. The architecture of a recurrent neural network [28].

Given a question pair r_{ij}, the n-dimensional word vector x_i corresponding to the ith word in r_{ij} is denoted as $x_i \in \mathbb{R}^n$. RNN consist of the input layer, hidden layer, and output layer. RNN are able to process a text sequence of arbitrary length by recursively applying a transition function to its internal hidden state vector h_i of the input sequence. The architecture of a Recurrent Neural Network is shown in Fig. 4. The hidden state h_i at the current time step i is calculated by the current input x_i and the previous hidden state h_{t-1}. If i is 0, then h_i is equal to 0; otherwise, h_i is defined as follows:

$$h_i = Relu(L \cdot h_{i-1} + E \cdot x_i + b), \tag{5}$$

where $Relu$ is a rectified linear function [29], and E is a weight vector, and L is a weight parameter, and b denotes a bias term. Thus, the hidden state h_m at the last step m is obtained by (5).

WV-RNN has a fully connected layer followed by a softmax non-linear layer that is used to predict the probability distribution P_r of r_{ij}. P_r is defined as follows:

$$P_r = softmax(U \cdot h_m + B), \tag{6}$$

where U is a weight vector, and B is a bias term.

To determine whether a duplicate question pair consists of a given question and a master question, a rank list of top K duplicate question pairs with the highest probability ranking is obtained.

E. WV-LSTM

Long Short-Term Memory (LSTM) [17] is a variant of RNN that is well suitable for processing sequential data. On the basis of RNN, LSTM introduces a gating mechanism to preserve long-term data dependencies on the basis of RNN. A WV-LSTM approach is constructed, which is based on Word2Vec and LSTM, to detect duplicate questions in Stack Overflow.

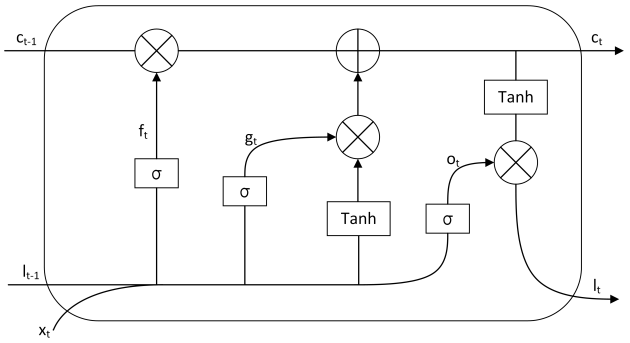

FIGURE 5. The architecture of an LSTM cell [10].

LSTM reads one element of a sequence at each time step t like RNN and calculates the output l_t by the input x_t and the previous output l_{t-1}. Fig. 5 shows the architecture of a typical LSTM cell. The cell state c_t of LSTM is controlled by three gates, which are a forget gate f_t, an input gate g_t, and an output gate o_t respectively. Each gate uses a sigmoid function $\sigma(y) = 1/(1 + e^y)$. The input of each gate is determined by the input of the current time step x_t and the output of the last time step l_{t-1}. Finally, the output l_t is computed by the cell state c_t and the output gate o_t. They are defined by as follows:

$$g_t = \sigma(W_{xg}x_t + W_{lg}l_{t-1} + b_g), \tag{7}$$
$$f_t = \sigma(W_{xf}x_t + W_{lf}l_{t-1} + b_f), \tag{8}$$
$$c_t = f_t c_{t-1} + g_t tanh(W_{xc}x_t + W_{lc}l_{t-1} + b_c), \tag{9}$$
$$o_t = \sigma(W_{xo}x_t + W_{lo}l_{t-1} + b_o), \tag{10}$$
$$l_t = o_t tanh(c_t), \tag{11}$$

where W is the weight vector, and b is the bias term, and $tanh$ is the hyperbolic tangent function.

WV-LSTM uses a softmax function to predict the probability distribution for each question pair. The probability distribution P_l of r_{ij} is defined as follows:

$$P_l = softmax(U \cdot l_t + B). \tag{12}$$

A rank list of the top K duplicate question pairs with the highest probability ranking is obtained, which is utilized to determine whether the master question of a given new question is detected.

IV. EVALUATION

In this section, we conduct experiments to evaluate the effectiveness of our approaches. In particular, we focus on three research questions (RQ):

RQ1: Which of our three approaches (i.e., WV-CNN, WV-RNN, and WV-LSTM) is superior to the four baseline approaches (i.e., DupPredictor, Dupe, DupPredictorRep-T, and DupeRep) for solving the problem of duplicate question detection?

Our three approaches WV-CNN, WV-RNN and WV-LSTM are based on Word2Vec, CNN, RNN and LSTM, which

are different from four baseline approaches DupPredictor, Dupe, DupPredictorRep-T, and DupeRep. The answer to this research question will indicate whether any of our three deep learning approaches can improve the effectiveness of duplicate question detection.

RQ2: Which of our three approaches WV-CNN, WV-RNN and WV-LSTM outperforms the four different machine learning approaches SVM, LR, RF and Xgboost in the duplicate question detection tasks?

Our three approaches WV-CNN, WV-RNN and WV-LSTM, which are based on Word2Vec, CNN, RNN and LSTM, are used to detect duplicate questions. As in our previous work [12], four different popular machine learning approaches are chosen to detect duplicate questions and compare them with our approaches. The chosen four machine learning approaches are SVM, LR, RF, and Xgboost respectively. The answer to this research question will demonstrate whether any of our three approaches are more effective than four machine learning approaches.

RQ3: Which of our three approaches WV-CNN, WV-RNN and WV-LSTM is better than DQ-CNN, DQ-RNN and DQ-LSTM for detecting duplicate questions.

Our three approaches (i.e., WV-CNN, WV-RNN, and WV-LSTM) are built based on Word2Vec, CNN, RNN and LSTM to detect duplicate questions. In our previous work [12], DQ-CNN, DQ-RNN and DQ-LSTM are constructed based on CNN, RNN and LSTM to detect duplicate questions. DQ-CNN, DQ-RNN and DQ-LSTM are compared with our approaches WV-CNN, WV-RNN and WV-LSTM. The answer to this research question will analyze whether our approaches can further improve the effectiveness of duplicate question detection.

A. DATASETS AND SETUP

Stack Overflow is a popular CQA website dedicated to software programming. The questions from Stack Overflow are collected as our experimental data. According to the data collection in Section II, 134,261 nonmaster questions and 88,476 master questions that are created from August 2008 to September 2014 in Stack Overflow are first extracted. Based on these questions, we collect the questions of six different question groups tagged with Java, Html, Python, C++, Ruby, and Objective-C respectively, as in previous work [4]. Then, duplicate question pairs and nonduplicate question pairs of six different question groups are built respectively. The datasets of six different question groups are shown in Table 1. For each question group, 80% of the duplicate question pairs are used as training data, and the remaining 20% of duplicate question pairs are used as testing data. Similarly, 80% of non-duplicate question pairs are used for training. Besides, all parameters of three deep learning models are set to their default values in our experiments. For example, the main parameters of CNN are in_channels and kernel_size, whose values are 1 and 5 respectively. The hidden_state and num_layers are the main parameters of RNN and LSTM, and their values are 64 and 2 respectively.

TABLE 1. The datasets of six different question groups.

Datasets	Master Question	Duplicate Question Pairs	Non-duplicate Question Pairs	All Question Pairs
Java	10198	17846	17846	35692
C++	6128	11039	11039	22078
Python	5302	8831	8831	17662
Ruby	1573	2084	2084	4168
Html	3617	6070	6070	12140
Objective-C	1852	4629	4629	9258

B. EVALUATION METRIC

To evaluate the performance of the compared approaches, $recall - rate@k$ was used in previous works [2]–[4]. And k is set to 5, 10, and 20 respectively. Thus, we also use these three metrics to evaluate our experimental results. The $recall - rate@k$ is defined as follows:

$$recall - rate@k = \frac{N_k}{N_{all}}, \tag{13}$$

where N_k is the total number of duplicate questions that have their corresponding master questions in the list of top K questions, and N_{all} is the total number of duplicate questions in our test data. The $recall - rate@k$ is used to measure the percentage of duplicate questions that successfully detected the master questions in the list.

According to previous work [30], test-retest reliability is used to evaluate the reliability of experimental results, and the Pearson correlation coefficient is used to measure the strength of correlation between the results obtained by performing the same experiment twice. $\rho_{X,Y}$ is presented as the Pearson correlation coefficient, which is defined as:

$$\rho_{X,Y} = \frac{\sum_{i=1}^{N_{all}}(X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^{N_{all}}(X_i - \bar{X})^2}\sqrt{\sum_{i=1}^{N_{all}}(Y_i - \bar{Y})^2}}, \tag{14}$$

where N_{all} is the total number of duplicate questions in our test data, and X_i is the prediction value of the i-th duplicate question in experimental results of the first group, and $\bar{X}$ is mean value in experimental results of the first group (i.e., $\bar{X} = \sum X_i/N_{all}$), and Y_i is the prediction value of the i-th duplicate question in experimental results of the second group, and $\bar{Y}$ is mean value in experimental results of the second group (i.e., $\bar{Y} = \sum Y_i/N_{all}$). If $0.9 \leq \rho_{X,Y} \leq 1$, the results have excellent reliability. If $0.8 \leq \rho_{X,Y} < 0.9$, the results have good reliability. If $0.7 \leq \rho_{X,Y} < 0.8$, the results have acceptable reliability. If $0.6 \leq \rho_{X,Y} < 0.7$, the results have questionable reliability. If $0.5 \leq \rho_{X,Y} < 0.6$, the results have poor reliability. If $\rho_{X,Y} < 0.5$, the results have unacceptable reliability.

C. BASELINE APPROACHES

In order to evaluate the performance of our approaches (i.e., WV-CNN, WV-RNN, and WV-LSTM), we describe the baseline approaches as follows:

DupPredictor: Previous work [3] proposed a DupPredictor approach to automatically detect duplicate questions in Stack Overflow, which took a new question as input and detected

potential duplicate questions of this question by considering multiple factors. They considered four factors that are the similarity scores of topics, titles, descriptions and tags. They first calculated similarity score between topics of each question pair by using a Latent Dirichlet Allocation (LDA) topic model. Then, the similarity score between titles of each question pair is computed by common words that they share. Next, the similarity score between descriptions of each question pair and the similarity score between tags of each question pair are calculated in the same way. Finally, they obtained a new similarity score by combining the similarity scores of the above four factors to detect duplicate questions in Stack Overflow.

Dupe: Previous work [2] proposed a Dupe approach to solve the problem of duplicate question detection in Stack Overflow, which combined five features obtained through the question pairs. These five features consisted of cosine similarity value, term overlap, entity overlap, entity type overlap, and WordNet similarity respectively. They first obtained the cosine similarity that calculates the cosine of the angle between a pair of questions based on the vector space model. Then, the term overlap was computed by using the textual similarity function. Next, they used the Jaccard coefficient to calculate entity overlap. In addition, entity type overlap was obtained by the overlapping of entities' types. Finally, the WordNet similarity was calculated by WS4J.

DupPredictorRep-T and DupeRep: Previous work [4] proposed two reproduction approaches based on DupPredictor [3] and Dupe [2], namely DupPredictorRep-T and DupeRep. DupPredictorRep-T was a reproduction approach of DupPredictor that did not consider the topic factor. They publicly provided a replication package so that other researchers could repeat, improve, or refute their results.

D. RESULTS

In this section, We first illustrate experimental results of compared baseline approaches and our approaches based on the datasets of the six different question groups. Next, we compare our approaches with four different popular machine learning approaches to analyze the effectiveness of our approaches. Finally, our approaches are compared with three deep learning approaches to illustrate the effectiveness of our approaches in the duplicate question detection tasks.

1) COMPARISON WITH BASELINE APPROACHES

Our three approaches WV-CNN, WV-RNN and WV-LSTM are built based on Word2Vec, CNN, RNN and LSTM to solve the problem of duplicate question detection in Stack Overflow. To evaluate the effectiveness of our approaches, four baseline approaches DupPredictor [3], Dupe [2], DupPredictorRep-T [4] and DupeRep [4] are used to compare with our approaches (i.e., WV-CNN, WV-RNN, and WV-LSTM).

Table 2 shows the experimental results of baseline approaches and our approaches in six different question groups. In Table 2, we use bold font to mark the best experimental results of six different question groups. "DupPredictorRep-T + Tag" means that DupPredictorRep-T is evaluated by the datasets of six different question groups tagged with Java, Html, Python, C++, Ruby, and Objective-C. We observe from the results of six different question groups that our approaches WV-CNN, WV-RNN and WV-LSTM outperform DupPredictor, Dupe, DupPredictorRep-T and DupeRep in terms of recall-rate@5, recall-rate@10. The reason is that our approaches fully capture word-level and document-level semantic information of each question pair. The experimental results show that our approaches WV-CNN and WV-LSTM are obviously better than DupPredictor, Dupe, DupPredictorRep-T and DupeRep in terms of recall-rate@5, recall-rate@10 and recall-rate@20. For example, in the Java question group, Dupe achieves recall-rate@5 of 38.25% whereas WV-CNN achieves recall-rate@5 of 81.27%, an increase of 112.47%. Similarity, we can see that WV-CNN has better results than baseline approaches in the C++, Python, Html and Objective-C question groups. Moreover, the results form six

TABLE 2. Recall-rate values of baseline approaches and our approaches.

Question Group	Technique	Recall-Rate (%)		
		Top-5	Top-10	Top-20
Java	DupPredictor	30.00	35.00	41.00
	Dupe	38.25	44.55	53.02
	DupPredictorRep-T + Tag	28.04	36.70	44.02
	DupeRep	19.44	23.90	27.95
	WV-CNN	81.27	81.27	81.30
	WV-RNN	65.17	65.20	65.25
	WV-LSTM	**82.06**	**82.15**	**82.20**
C++	DupPredictor	26.00	28.00	35.00
	Dupe	37.14	40.12	49.93
	DupPredictorRep-T + Tag	18.19	25.91	33.73
	DupeRep	16.26	20.63	25.67
	WV-CNN	80.01	80.06	80.10
	WV-RNN	60.16	60.20	60.25
	WV-LSTM	**80.15**	**80.19**	**80.28**
Python	DupPredictor	28.15	32.28	38.74
	Dupe	37.20	45.41	53.22
	DupPredictorRep-T + Tag	0.00	22.61	34.38
	DupeRep	16.38	20.00	24.72
	WV-CNN	79.50	79.67	79.84
	WV-RNN	58.36	58.48	58.65
	WV-LSTM	**79.61**	**79.78**	**80.01**
Ruby	DupPredictor	33.00	37.00	39.00
	Dupe	51.16	59.56	66.11
	DupPredictorRep-T + Tag	17.31	30.99	39.47
	DupeRep	29.84	35.45	38.26
	WV-CNN	76.76	77.00	77.72
	WV-RNN	61.26	61.50	62.71
	WV-LSTM	**76.76**	**77.24**	**77.97**
Html	DupPredictor	25.00	28.00	34.00
	Dupe	37.14	39.45	50.23
	DupPredictorRep-T + Tag	28.21	35.89	43.56
	DupeRep	18.15	21.67	25.30
	WV-CNN	81.41	81.49	81.66
	WV-RNN	67.47	67.63	67.72
	WV-LSTM	**81.58**	**81.74**	**81.91**
Objective-C	DupPredictor	26.00	28.00	31.00
	Dupe	40.61	47.88	56.35
	DupPredictorRep-T + Tag	29.52	33.85	39.37
	DupeRep	23.62	30.64	38.10
	WV-CNN	75.90	76.11	76.22
	WV-RNN	55.70	55.81	55.92
	WV-LSTM	**78.39**	**78.61**	**78.94**

different question groups show that WV-RNN has a good improvement over DupPredictor, Dupe, DupPredictorRep-T and DupeRep in terms of recall-rate@5, recall-rate@10 and recall-rate@20, except for the Ruby and Objective-C question groups. Therefore, our approaches are more effective than baseline approaches to the duplicate question detection tasks in Stack Overflow.

RQ1: The experimental results from six different question groups show that WV-CNN, WV-RNN and WV-LSTM, are based on Word2Vec, CNN, RNN and LSTM, outperform the four baseline approaches (i.e., DupPredictor, Dupe, DupPredictorRep-T and DupeRep) in terms of recall-rate@5, recall-rate@10.

2) COMPARISON WITH MACHINE LEARNING APPROACHES

Our three approaches WV-CNN, WV-RNN and WV-LSTM are based on Word2Vec, CNN, RNN and LSTM and are used to detect duplicate questions in Stack Overflow. WV-CNN, WV-RNN and WV-LSTM are compared with four different machine learning approaches (i.e., SVM, LR, RF, and Xgboost) to evaluate the effectiveness of our approaches in duplicate question detection tasks.

The experimental results of four machine learning approaches (i.e., SVM, LR, RF, and Xgboost) and our three approaches (i.e., WV-CNN, WV-RNN, and WV-LSTM) are shown in Table 3. For six different question groups, the experimental results show that WV-CNN and WV-LSTM are obviously better than SVM, LR, RF and Xgboost in terms of recall-rate@5, recall-rate@10 and recall-rate@20. In Table 3, we mark the best experimental results of six different question groups by using bold fold. Moreover, in terms of recall-rate@5, recall-rate@10 and recall-rate@20, the results from six different question groups indicate that WV-RNN has better performance than SVM, LR, RF and Xgboost. For example, in the C++ question group, Xgboost achieves recall-rate@5 of 52.76% whereas WV-LSTM achieves recall-rate@5 of 80.15%, an increase of 51.92%. Therefore, the experimental results from six different question groups demonstrate that our approaches (i.e., WV-CNN, WV-RNN, and WV-LSTM) are superior to four machine learning approaches (i.e., SVM, LR, RF, and Xgboost) in terms of recall-rate@5, recall-rate@10 and recall-rate@20.

RQ2: The experimental results illustrate that WV-CNN, WV-RNN and WV-LSTM, which are based on Word2Vec, CNN, RNN and LSTM, outperform the four machine learning approaches SVM, LR, RF and Xgboost for six different question groups in terms of recall-rate@5, recall-rate@10 and recall-rate@20.

3) COMPARISON WITH DEEP LEARNING APPROACHES

Our three approaches WV-CNN, WV-RNN and WV-LSTM that are based on Word2Vec, CNN, RNN and LSTM are used to detect duplicate questions in Stack Overflow. Due to Word2Vec can fully capture the semantic information at word-level, we use Word2Vec to obtain the word vector representations of each question pair in our approaches. To demonstrate the effectiveness of using Word2Vec in our approaches, three deep learning approaches (i.e., DQ-CNN, DQ-RNN, and DQ-LSTM) are used to compare with our three approaches (i.e., WV-CNN, WV-RNN, and WV-LSTM).

The experimental results of three deep learning approaches and our approaches are shown in Table 4. In Table 4, the best experimental results of six different question groups are marked by using bold font. From this table, we can also observe from the results of six different question groups that WV-CNN and WV-LSTM have good improvements over DQ-CNN, DQ-RNN and DQ-LSTM in terms of recall-rate@5, recall-rate@10 and recall-rate@20. For instance, in the Python question group, DQ-LSTM achieves recall-rate@20 of 57.04% whereas WV-LSTM achieves recall-rate@20 of 80.01%, an increase of 40.97%. We can also

TABLE 3. Recall-rate values of machine learning approaches and our approaches.

Question Group	Technique	Recall-Rate (%)		
		Top-5	Top-10	Top-20
Java	SVM	50.30	50.34	50.45
	LR	52.74	52.77	53.05
	RF	46.95	46.98	47.18
	Xgboost	53.79	53.90	53.98
	WV-CNN	81.27	81.27	81.30
	WV-RNN	65.17	65.20	65.25
	WV-LSTM	**82.06**	**82.15**	**82.20**
C++	SVM	51.65	51.75	51.79
	LR	49.82	50.09	50.37
	RF	47.70	47.75	47.93
	Xgboost	52.76	52.90	53.08
	WV-CNN	80.01	80.06	80.10
	WV-RNN	60.16	60.20	60.25
	WV-LSTM	**80.15**	**80.19**	**80.28**
Python	SVM	56.47	56.59	56.99
	LR	56.30	56.41	56.82
	RF	54.41	54.58	55.15
	Xgboost	56.41	56.53	56.70
	WV-CNN	79.50	79.67	79.84
	WV-RNN	58.36	58.48	58.65
	WV-LSTM	**79.61**	**79.78**	**80.01**
Ruby	SVM	51.82	52.06	53.27
	LR	53.75	54.96	56.17
	RF	40.19	41.40	43.10
	Xgboost	54.00	54.72	56.42
	WV-CNN	76.76	77.00	77.72
	WV-RNN	61.26	61.50	62.71
	WV-LSTM	**76.76**	**77.24**	**77.97**
Html	SVM	58.59	58.76	59.17
	LR	56.68	56.93	57.26
	RF	52.03	52.20	53.20
	Xgboost	54.11	54.36	54.85
	WV-CNN	81.41	81.49	81.66
	WV-RNN	67.47	67.63	67.72
	WV-LSTM	**81.58**	**81.74**	**81.91**
Objective-C	SVM	41.59	41.69	41.80
	LR	55.05	55.27	55.59
	RF	41.15	41.48	41.59
	Xgboost	54.51	54.61	55.16
	WV-CNN	75.90	76.11	76.22
	WV-RNN	55.70	55.81	55.92
	WV-LSTM	**78.39**	**78.61**	**78.94**

TABLE 4. Recall-rate values of deep learning approaches and our approaches.

Question Group	Technique	Recall-Rate (%)		
		Top-5	Top-10	Top-20
Java	DQ-CNN	56.69	56.78	56.98
	DQ-RNN	52.99	53.05	53.16
	DQ-LSTM	59.35	59.41	59.52
	WV-CNN	81.27	81.27	81.30
	WV-RNN	65.17	65.20	65.25
	WV-LSTM	**82.06**	**82.15**	**82.20**
C++	DQ-CNN	60.06	60.20	60.39
	DQ-RNN	59.51	59.56	59.88
	DQ-LSTM	59.93	60.11	60.48
	WV-CNN	80.01	80.06	80.10
	WV-RNN	60.16	60.20	60.25
	WV-LSTM	**80.15**	**80.19**	**80.28**
Python	DQ-CNN	56.53	56.76	56.99
	DQ-RNN	55.84	55.96	56.13
	DQ-LSTM	56.47	56.76	57.04
	WV-CNN	79.50	79.67	79.84
	WV-RNN	58.36	58.48	58.65
	WV-LSTM	**79.61**	**79.78**	**80.01**
Ruby	DQ-CNN	54.24	55.21	57.87
	DQ-RNN	52.78	53.75	55.21
	DQ-LSTM	54.96	55.45	56.42
	Word2Vec-CNN	76.76	77.00	77.72
	Word2Vec-RNN	61.26	61.50	62.71
	Word2Vec-LSTM	**76.76**	**77.24**	**77.97**
Html	DQ-CNN	58.76	58.84	59.34
	DQ-RNN	54.94	55.27	56.02
	DQ-LSTM	58.84	59.34	59.75
	WV-CNN	81.41	81.49	81.66
	WV-RNN	67.47	67.63	67.72
	WV-LSTM	**81.58**	**81.74**	**81.91**
Objective-C	DQ-CNN	55.16	55.59	55.70
	DQ-RNN	54.07	54.18	54.61
	DQ-LSTM	55.81	55.92	56.68
	WV-CNN	75.90	76.11	76.22
	WV-RNN	55.70	55.81	55.92
	WV-LSTM	**78.39**	**78.61**	**78.94**

see that WV-LSTM has similar results for other question groups tagged with Java, C++, Ruby, Html, and Objective-C. Furthermore, the experimental results from six different question groups show that our approach WV-RNN outperforms DQ-CNN, DQ-RNN and DQ-LSTM in terms of recall-rate@5, recall-rate@10 and recall-rate@20, except for the C++ and Objective-C question groups. We find that word-level semantics obtained by using Word2Vec are especially useful for detecting duplicate questions in Stack Overflow. Therefore, our approaches are effective for duplicate question detection tasks in Stack Overflow.

Furthermore, we use the test-retest reliability approach to evaluate the reliability of our results. We repeat two tests for our three approaches and compare the effects of the two tests. The evaluation results of reliability are shown in Table 5. From Table 5, we can see that the values of Pearson correlation coefficient from six different question groups are greater than 0.8, which shows that our experimental results have good reliability. In particular, the values of Pearson correlation coefficient of our approaches are greater than 0.9 in most cases. This suggests that the most of experimental results of our approaches have the excellent reliability.

TABLE 5. Evaluation results of reliability.

Question Group	Technique	Pearson		
		Top-5	Top-10	Top-20
Java	WV-CNN	0.98	0.97	0.97
	WV-RNN	0.90	0.90	0.89
	WV-LSTM	0.99	0.99	0.98
C++	WV-CNN	0.91	0.90	0.89
	WV-RNN	0.90	0.89	0.88
	WV-LSTM	1.00	0.99	0.99
Python	WV-CNN	0.99	0.98	0.98
	WV-RNN	0.90	0.90	0.89
	WV-LSTM	0.96	0.95	0.94
Ruby	WV-CNN	0.95	0.91	0.86
	WV-RNN	0.94	0.92	0.91
	WV-LSTM	0.97	0.92	0.87
Html	WV-CNN	0.98	0.97	0.96
	WV-RNN	0.94	0.93	0.92
	WV-LSTM	0.97	0.97	0.96
Objective-C	WV-CNN	0.90	0.88	0.86
	WV-RNN	0.89	0.88	0.87
	WV-LSTM	0.99	0.99	0.96

RQ3: The experimental results from six different question groups demonstrate that WV-CNN and WV-LSTM, which are based on Word2Vec, CNN and LSTM, are better than three deep learning approaches (i.e., DQ-CNN, DQ-RNN, and DQ-LSTM) in terms of recall-rate@5, recall-rate@10 and recall-rate@20.

V. THREATS TO VALIDITY

There are several threats that may potentially affect the validity of our experiments. In this section, we introduce the threats to validity of our experiments and how we alleviate these threats.

A. CONSTRUCT VALIDITY

The construct validity refers to the correct identification of measures used in the measurement procedure. The recall-rate@k metric is used to evaluate the effectiveness of our approaches on the duplicate question detection tasks, and this metric has also been utilized to detect the effectiveness of duplicate questions in previous works [2]–[4]. Therefore, we believe that there is little threat to construct validity.

B. INTERNAL VALIDITY

The internal validity refers to experimenter bias and errors in our experiments. We analyze the datasets of six different question groups, and all datasets are collected from Stack Overflow. We have double checked our experimental data and evaluate reliability of our results by using test-retest reliability approach. Moreover, the datasets of the six different question groups are also used in previous work [2], [4].

C. EXTERNAL VALIDITY

Our approaches are evaluated by the real data in Stack Overflow, which may not guarantee the generalization of

our approaches. To alleviate this threat, we use the datasets of six different question groups based on six different popular programming languages as our experiment data. In future work, our approaches will be applied to other CQA websites.

VI. RELATED WORK

In this section, we describe previous studies related to our work, which include duplicate question detection and duplicate bug reports detection.

A. DUPLICATE QUESTION DETECTION

There are some duplicate question detection approaches that have been proposed [2]–[4], [27], [31]–[33]. Manku *et al.* presented a Charikar's fingerprinting technique for the near-duplicate web page detection [31]. Hajishirzi *et al.* proposed an adaptive near-duplicate detection approach to detect a near-duplicate document by similarity learning [32]. Tao *et al.* developed a framework based on syntactical characteristics, semantic similarity and contextual information for duplicate twitters detection [33]. Bogdanova *et al.* proposed a CNN-based approach to identify semantically equivalent questions in the CQA website dedicated to Ask Ubuntu [27]. Zhang *et al.* proposed a DupPredictor approach considering multiple factors to detect duplicate questions in Stack Overflow. These factors are the similarity scores of topics, titles, descriptions and tags on each question pair. They first calculated the similarity score between topics of each question pair by LDA topic model. Then, the similarity scores of titles, descriptions and tags for each question pair are computed by their common words respectively. Finally, a new similarity score is obtained by combining these four similarity scores and used to detect duplicate questions in Stack Overflow [3]. Ahasanuzzaman *et al.* proposed a Dupe approach based on cosine similarity value, term overlap, entity overlap, entity type overlap and wordNet similarity to solve the problem of duplicate question detection in Stack Overflow [2]. Silva *et al.* [4] proposed two reproduction approaches DupPredictorRep-T and DupRep for DupPredictor [3] and Dupe [2] to detect duplicate questions in Stack Overflow. Different from the above approaches, our approaches are based on three different deep learning techniques and Word2Vec to detect duplicate questions in Stack Overflow, which can fully capture the document-level and word-level semantic information respectively.

B. DUPLICATE BUG REPORTS DETECTION

Some previous studies are proposed on duplicate bug reports detection [8], [34]–[38]. Runeson *et al.* developed a prototype tool by natural language processing techniques to automatically detect duplicate bug reports [34]. Wang *et al.* proposed an approach based on natural language information and execution information to detect duplicate bug reports [35]. Sun *et al.* used the discriminative models for information retrieval to detect duplicate bug reports [36]. Sun *et al.* proposed an approach by extending BM25F to retrieve the duplicate bug reports, and also presented a retrieval function (REP) to evaluate the similarity between two bug reports [37]. Yang *et al.* proposed an approach based on a traditional information retrieval technique and a word embedding technique to recommend similar bugs [38]. This approach consists of four factors, namely titles, descriptions, product information, and component information. Xie *et al.* explored the use of word embedding and CNN to compute the similarity between the pairs bug report and to identify possible duplicate bug reports [8]. However, these approaches focus on the problem of duplicate bug reports detection. Different from the above approaches, our approaches are used to solve the problem of duplicate question detection in Stack Overflow.

VII. CONCLUSION

With the widespread application of deep learning, we explore the use of deep learning techniques and Word2Vec to detect duplicate questions in Stack Overflow. Three different deep learning techniques are considered to solve the problem of duplicate question detection in Stack Overflow, such as CNN, RNN, and LSTM. Moreover, Word2Vec is used to obtain the vector representations of words. In this paper, three deep leaning approaches WV-CNN, WV-RNN and WV-LSTM are built based on Word2Vec, CNN, RNN and LSTM to detect duplicate questions in Stack Overflow. They can fully capture the word-level and document-level semantic information of each question pair in Stack Overflow. The experimental results from six different question groups show that our approaches WV-CNN and WV-LSTM are obviously better than four baseline approaches (i.e, DupPredictor, Dupe, DupePredictorRep-T, and DupeRep) and four machine learning approaches (i.e., SVM, LR, RF, and Xgboost) in terms of recall-rate@5, recall-rate@10 and recall-rate@20. Furthermore, the results of six different question groups indicate that our approaches WV-CNN, WV-RNN and WV-LSTM have good improvements over three deep learning approaches (i.e., DQ-CNN, DQ-RNN, and DQ-LSTM) in terms of recall-rate@5, recall-rate@10 and recall-rate@20.

REFERENCES

[1] D. Correa and A. Sureka, "Chaff from the wheat: Characterization and modeling of deleted questions on stack overflow," in *Proc. 23rd Int. Conf. World Wide Web (WWW)*, Jan. 2014, pp. 631–642.
[2] M. Ahasanuzzaman, M. Asaduzzaman, C. K. Roy, and K. A. Schneider, "Mining duplicate questions in stack overflow," in *Proc. 13th Min. Softw. Repositories (MSR)*, May 2016, pp. 402–412.
[3] Y. Zhang, D. Lo, X. Xia, and J.-L. Sun, "Multi-factor duplicate question detection in stack overflow," *J. Comput. Sci. Technol.*, vol. 30, no. 5, pp. 981–997, Sep. 2015.
[4] R. F. Silva, K. Paixão, and M. de Almeida Maia, "Duplicate question detection in stack overflow: A reproducibility study," in *Proc. 25th Softw. An. Ev. Reeng. (SANER)*, Mar. 2018, pp. 572–581.
[5] C. N. Kamath, S. S. Bukhari, and A. Dengel, "Comparative study between traditional machine learning and deep learning approaches for text classification," in *Proc. ACM Symp. Document Eng.*, Halifax, NS, Canada, 2018, pp. 1–11.
[6] J. Kapočiūtė-Dzikienė, R. Damaševičius, and M. Wozniak, "Sentiment analysis of lithuanian texts using traditional and deep learning approaches," *Computers*, vol. 8, no. 1, pp. 1–16, Jan. 2019.
[7] M. White, M. Tufano, C. Vendome, and D. Poshyvanyk, "Deep learning code fragments for code clone detection," in *Proc. 31st IEEE/ACM Int. Conf. Automat. Softw. Eng. (ASE)*, Sep. 2016, pp. 87–98.

[8] Q. Xie, Z. Wen, J. Zhu, C. Gao, and Z. Zheng, "Detecting duplicate bug reports with convolutional neural networks," in *Proc. 25th Asia–Pacific Softw. Eng. Conf. (APSEC)*, Dec. 2018, pp. 416–425.
[9] B. Xu, D. Ye, Z. Xing, X. Xia, G. Chen, and S. Li, "Predicting semantically linkable knowledge in developer online forums via convolutional neural network," in *Proc. 31st IEEE/ACM Int. Conf. Autom. Softw. Eng. (ASE)*, Sep. 2016, pp. 51–62.
[10] H. Liang, Y. Yu, L. Jiang, and Z. Xie, "Seml: A semantic LSTM model for software defect prediction," *IEEE Access*, vol. 7, pp. 83812–83824, 2019.
[11] V. J. Hellendoorn and P. Devanbu, "Are deep neural networks the best choice for modeling source code?" in *Proc. 11th Joint Meeting Found. Softw. Eng. (FSE)*, Sep. 2017, pp. 763–773.
[12] L.-T. Wang, L. Zhang, and J. Jiang, "Detecting Duplicate Questions in Stack Overflow via Deep Learning Approaches," presented at the 26th Asia–Pacific Softw. Eng. Conf. (APSEC), Putrajaya, Malaysia, Dec. 2019.
[13] T. Mikolov, K. Chen, G. Gorrado, and J. Dean, "Efficient estimation of word representations in vector space," 2013, *arXiv:1301.3781*. [Online]. Available: https://arxiv.org/abs/1301.3781
[14] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proc. 26th Int. Conf. Neural Inf. Process. Syst.*, Oct. 2013, pp. 3111–3119.
[15] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. 25th Int. Conf. Neural Inf. Process. Syst.*, Jun. 2012, pp. 1097–1105.
[16] A. Graves, A. R. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, May 2013, pp. 6645–6649.
[17] T. N. Sainath, O. Vinyals, A. Senior, and H. Sak, "Convolutional, long short-term memory, fully connected deep neural networks," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, Apr. 2015, pp. 4580–4584.
[18] T. Joachims, "Training linear SVMs in linear time," in *Proc. 12th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Philadelphia, PA, USA, 2006, pp. 217–226.
[19] A. Genkin, D. D. Lewis, and D. Madigan, "Large-scale Bayesian logistic regression for text categorization," *Technometrics*, vol. 49, no. 3, pp. 291–304, Aug. 2007.
[20] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, Oct. 2001.
[21] T. Chen and T. He, "XGBoost: Extreme gradient boosting," in *R Package Version 0.4-2*. 2015, pp. 1–4.
[22] F. Brooks and H. Kugler, "No silver bullet," *IEEE Comput.*, vol. 20, no. 4, pp. 10–19, 1987.
[23] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, "A neural probabilistic language model," *J. Mach. Learn. Res.*, vol. 3, pp. 1137–1155, Feb. 2003.
[24] H. Schwenk, "Continuous space language models," *Comput. Speech Lang.*, vol. 21, no. 3, pp. 492–518, Jul. 2007.
[25] S. Deerwester, S. T. Dumais, G. W. Fumas, T. K. Landauer, and R. Harshman, "Indexing by latent semantic analysis," *J. Amer. Soc. Inf. Sci.*, vol. 41, pp. 391–407, Sep. 1990.
[26] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent Dirichlet allocation," *J. Mach. Learn. Res.*, vol. 3, pp. 993–1022, Jan. 2003.
[27] D. Bogdanova, C. dos Santos, L. Barbosa, and B. Zadrozny, "Detecting semantically equivalent questions in online user forums," in *Proc. 19th Comput. Natural Lang. Learn.*, Jul. 2015, pp. 123–131.
[28] P. Zhou, J. Liu, X. Liu, Z. Yang, and J. Grundy, "Is deep learning better than traditional approaches in tag recommendation for software information sites?" *Inf. Softw. Technol.*, vol. 109, pp. 1–13, May 2019.
[29] G. E. Dahl, T. N. Sainath, and G. E. Hinton, "Improving deep neural networks for LVCSR using rectified linear units and dropout," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, May 2013, pp. 8609–8613.
[30] L. Guttman, "A basis for analyzing test-retest reliability," *Psychometrika*, vol. 10, no. 4, pp. 255–282, Dec. 1945.
[31] G. S. Manku, A. Jain, and A. D. Sarma, "Detecting near-duplicates for Web crawling," in *Proc. 16th World Wide Web (WWW)*, May 2007, pp. 141–150
[32] H. Hajishirzi, W. Yih, and A. Kolcz, "Adaptive near-duplicate detection via similarity learning," in *Proc. 33rd Int. ACM SIGIR Conf. R D Inf. Retr.*, Jul. 2010, pp. 419–426.
[33] K. Tao, F. Abel, C. Hauff, G. Houben, and U. Gadiraju, "Groundhog day: Near-duplicate detection on Twitter," in *Proc. 22nd Int. Conf. World Wide Web (WWW)*, May 2013, pp. 1273–1284.
[34] P. Runeson, M. Alexandersson, and O. Nyholm, "Detection of duplicate defect reports using natural language processing," in *Proc. 29th Int. Conf. Softw. Eng. (ICSE)*, May 2007, pp. 499–510.
[35] X. Wang, L. Zhang, T. Xie, J. Anvik, and J. Sun, "An approach to detecting duplicate bug reports using natural language and execution information," in *Proc. 30th Int. Conf. Softw. Eng. (ICSE)*, Dec. 2008, pp. 461–470.
[36] C. Sun, D. Lo, X. Wang, J. Jiang, and S. C. Khoo, "A discriminative model approach for accurate duplicate bug report retrieval," in *Proc. 32nd Int. Conf. Softw. Eng. (ICSE)*, May 2010, pp. 45–54.
[37] C. Sun, D. Lo, S. C. Khoo, and J. Jiang, "Towards more accurate retrieval of duplicate bug reports," in *Proc. 26th IEEE/ACM Int. Conf. Autom. Softw. Eng. (ASE)*, Nov. 2011, pp. 253–262.
[38] X. Yang, D. Lo, X. Xia, L. Bao, and J. Sun, "Combining word embedding with information retrieval to recommend similar bug reports," in *Proc. 26th Int. Symp. Softw. Reliab. Eng. (ISSRE)*, 2016, pp. 127–137.

LITING WANG received the master's degree in computer science and technology from Inner Mongolia University, Hohhot, China, in 2014. She is currently pursuing the Ph.D. degree with the School of Computer Science and Engineering, Beihang University, Beijing, China. Her research interests include software engineering, data mining, and deep learning.

LI ZHANG received the B.S., M.S., and Ph.D. degrees from the School of Computer Science and Engineering, Beihang University, Beijing, China, in 1989, 1992, and 1996, respectively. She is currently a Professor with the School of Computer Science and Engineering, Beihang University, where she is leading the expertise areas of system and software modeling. She has around 20 years of experience of conducting industry-oriented research in various application domains such as avionics and ships, and communications in several countries, including America, Norway, and France. Her current research interests include software engineering, with specific interest in requirements engineering, software/system architectures, model-based engineering, model-based product line engineering, and empirical software engineering.

JING JIANG received the B.S. and Ph.D. degrees in computer science from Peking University, Beijing, China, in 2007 and 2012, respectively. She is currently an Assistant Professor with the State Key Laboratory of Software Development Environment, Beihang University, Beijing. Her research interests include software engineering, empirical software engineering, data mining, and recommendation.