A Model of Convolutional Neural Network Combined with External Knowledge to Measure the Question Similarity for Community Question Answering Systems

Van-Tu Nguyen, Anh-Cuong Le, and Ha-Nam Nguyen

***Abstract*—Automatically determining similar questions and ranking the obtained questions according to their similarities to each input question is a very important task to any community Question Answering system (cQA). Various methods have applied for this task including conventional machine learning methods with feature extraction and some recent studies using deep learning methods. This paper addresses the problem of how to combine advantages of different methods into one unified model. Moreover, deep learning models are usually only effective for large data, while training data sets in cQA problems are often small, so the idea of integrating external knowledge into deep learning models for this cQA problem becomes more important. To this objective, we propose a neural network-based model which combines a Convolutional Neural Network (CNN) with features from other methods so that the deep learning model is enhanced with addtional knowledge sources. In our proposed model, the CNN component will learn the representation of two given questions, then combined additional features through a Multilayer Perceptron (MLP) to measure similarity between the two questions. We tested our proposed model on the SemEval 2016 task-3 data set and obtain better results in comparison with previous studies on the same task.**

***Index Terms*—Community based question answering, convolutional neural networks, combining multiple sources.**

I. Introduction

Nowadays, many cQA forums are becoming more and more popular and really useful such as StackOverflow[1] and Quora[2]. These systems contain millions of questions and corresponding answers created by cQA users. The questions and answers on these cQA forums are diverse and enable different users to find answers directly from complex and heterogeneous information. It is a natural way that whenever a cQA system receives a question, it firstly determine whether similar questions have existed or not, and if yes the system prefers to show these related question-answers contained in its database before waiting for new answers from other users. Therefore building a module for measuring the similarity between questions becomes an essential task in every cQA system.

In previous studies, particularly the conventional methods, the task of measuring similarity between two sentences is based on features extracted from linguistic analysis methods. These features are usually n-grams [1], [2] or richer linguistic information which requires deep analysis such as syntactic parsing [3]-[5]. The similarity degree is then computed based on some measures between two feature vectors such as Euclidean, Cosine, or Jaccard.

It has been widely shown that machine learning methods are applied successfully for most of the artificial intelligence problems. To address the problem of measuring question similarity we can formulate it as a binary classification problem (with the two labels standing for similar and not similar), and can apply any machine learning classification methods such as Support Vector Machines, Naive Bayesian classification, etc. Recently, some deep learning models such as CNN, Long Short-Term Memory (LSTM) have been shown very effective in many classification problems, and also for similarity measurement problems such as [6], [7]. However, from our observation, such kinds of studies for the particular task of question similarity measurement seems absent.

Actually, deep learning makes advantages because it has the ability to automatically learning abstract features via different layers of the deep neural models. However, as the major characteristic of statistical learning, such models are just efficient when the training data are large enough, especially for deep neural network-based models. It is also not clear whether a deep neural network contains within its internal structure other kinds of information that can be learned from other models. Therefore it should be remarkable that deep neural networks can get some complementary information from other models.

For the above observation, in this paper, we address the problem to utilize different methods and different information sources for improving the accuracy of measuring question similarity as well as ranking the similar questions with respect to an input question. To this objective, we firstly based on CNN, a very successful deep learning model, to formulate the problem of measuring the similarity between two questions. And then we extend this model to include additional information from other sources obtained from other models. Various kinds of additional information have been used including word2vec representation which represents a word as a vector of real numbers; linguistic features such as words and name entities; question types and question categories, which are obtained by classification. By these kinds of additional information with features derived

Manuscript received February 6, 2020; revised September 8, 2020.

Van-Tu Nguyen and Ha-Nam Nguyen are with the VNU University of Engineering and Technology, Ha Noi City, Vietnam (e-mail: tuspttb@gmail.com, namnh@vnu.edu.vn).

Anh-Cuong Le is with the Natural Language Processing and Knowledge Discovery Laboratory, Faculty of Information Technology, Ton Duc Thang University, Ho Chi Minh City, Vietnam (e-mail: leanhcuong@tdtu.edu.vn).

[1] https://stackoverflow.com/

[2] https://www.quora.com/

from the CNN component we finally generate the joint representation containing miscellaneous features. In another way, we can imagine that this model is an effective way of enhancing a deep learning model by providing complimentary additional knowledge, especially in the case of lacking training data. The joint representation is then put into a classification such as a multi-layer neural network which is well known as the fully connected layers for the classification stage.

To test the proposed idea, we conducted our experiments on the popular dataset for cQA which is the SemEval 2016 task 3, and implement different combinations of feature data sets to find out the best model.

The rest of the paper is organized as follows. Section II is related work. The main parts of this paper include: section III presents the CNN model for question representation and then for measuring similarity between two questions; Section IV presents different external knowledge sources and how to gain them; Section V is the important part in which we show how to integrate the external knowledge features into the CNN model. After completing the proposed model in theorical aspect, we present the data set and our experimental results in Section VI. Finally, the conclusion is shown in Section VII.

II. Related Work

Whenever cQA receives an input question, it needs to search its database for similar questions. The same set of questions is then ranked and used to extract possible answers to the input questions. However, determining the similarity between questions is still one of the major challenges in cQA due to problems such as “lexical gap”. Many different approaches have been proposed to overcome this problem.

Some early methods based on the statistical machine translation approach to computing the similarity between two questions. For example, in [8], [9] the authors used a translation model to compare questions. In [8], the authors constructed their translation models from a collection of predefined similar questions. In the study from [10], the authors rely on a machine translation model to find similar questions, in which the authors used information from both questions and answers. Although these machine translation based methods have shown encouraging results, they require a lot of labeled data to estimate the parameters, which is not easily achieved.

Some other studies have tried to go beyond the simple text representation of questions as presented in [11]-[15]. These studies used the data from Yahoo! Answer, in which the similarity question pairs are assigned by the user, sometimes assigned automatically based on some heuristic rules. In [11], the authors estimated a similarity measure using the category structure of the Yahoo! Answers, in which they identify a category of input question (e.g., travel, politics, or education) and then rank the questions stored in the cQA belonged to user's input question. In [12], [14], [15], the authors used the LDA (Latent Dirichlet Allocation) model to explore latent semantic topics, create question-answer pairs, and then used distributed learning topics to get similar questions. These LDA based methods have demonstrated that their models are significantly better than other models learned from questions, answers, or both in a simple "plus" way with traditional methods. In the study in [13], given a new question and a set of good candidate questions of similarity, the authors solved this problem in two steps: firstly these candidate questions are graphically represented by thematic terms, and secondly, they are ranked based on the graph.

Other studies use the results of the syntactic analysis as the main information for determining question similarities. The authors in [3] determined related questions to a new question by calculating the similarity between the syntactic trees of two questions. They used the tree similarity calculated based on the number of substructures shared between the two trees. The study in [2], the authors also used parsed trees, but the difference from [3] is that they used parsed trees directly in a tree kernel, with the use of the Kelp platform [16]. The method in [2] was applied to SemEval 2016 task 3 and shown results in [17]. The best performing system in this SemEval 2016 task 3 was presented in [18], in which they used SVMrank (Joachims, 2006) to optimize ranking and use various kinds of features including lexical-based and semantic-based features. In this study, the semantic features are achieved by using the word distribution representation, and the knowledge graph is constructed using the largest multilingual language network BabelNet from the FrameNet vocabulary database. However, because the data in the cQA is often noisy, sparse, and ambiguous, so the syntactic analysis often results in low efficiency. Therefore, adding lexical-based and semantic-based features certainly improve system’s performance, but they require complex semantic analysis on question sentences.

In recent years, deep neural network-based models are useful for machine learning [19]. They have been particularly successful in speech processing and computer vision tasks. Recently, deep learning methods have begun to overcome traditional, sparse linear models in Natural Language Processing (NLP) [20], [21]. Many recent studies have shown the effectiveness of deep neural network-based models for tasks such as answer selection [22], [23], sequence labeling [24], ranking question [25], and answer sentence selection [26]. In [25], the authors used a CNN model and bag-of-words (BOW) representations of input questions and related questions to estimate cosine similarity scores. In [27], the authors present a model based on LSTM and BOW representations of questions and their answers to assess the degree of relevance between them. In the study in [28], the authors proposed use of three different classifiers (naive Bayes, SVM, CNN), in which they combine scores of the three classifiers to rank the questions. If at least 2 out of 3 classifiers result in "related" then the new question is considered similar to the question in the database. Although the use of CNN-based models has shown impressed effectiveness in computer vision tasks and many NLP tasks but in this task of question similarity measurement for cQA it has not achieved the desired results. In our observations, the deep learning model (e.g. CNN) applied for cQA is currently showing limitation results due to the lack of annotated data. So our new approach is to integrate other information sources, which do not require the very large

data like deep learning methods, into the CNN model to overcome its limitation.

III. Modeling CNN for Question Similarity Measurement

CNN is known as one of the most successful deep learning models in computer vision and it recently attracts many studies in solving NLP problems such as machine translation [29], text classification [30], semantic analysis [31], search query retrieval [32], sentence modeling [20]. Actually, in computer vision, the input represented as a matrix of pixels that can be appropriately exploited by CNN operators but it is not straight when applying for NLP problems. In this section, we present our CNN based formulation for question similarity measurement.

A. General CNN Architecture

In a vanilla CNN model, there are sequentially layers including the convolutional layer, the pooling layer, and the fully connected layer. CNN receives a matrix of real numbers representing the input source, as shown in Fig. 1.

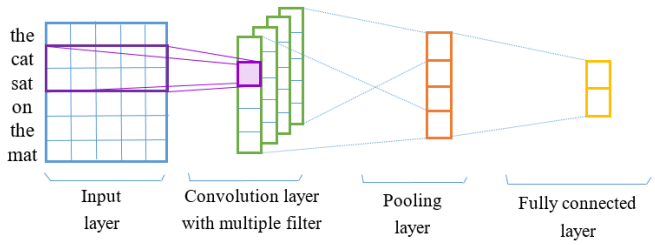

Fig. 1. A general CNN architecture.

Convolutional layer + ReLU

At the first stage, the input matrix is transferred via the convolutional layer with different filters for extracting synthesis features of the. Each filter is defined as a matrix and it slides from left to right and up to down through all the input matrix to get parts from the input matrix, which we call a slid matrix. Consequently, it generates the called convolved matrix via this filtering operator. Note that each filter may stand for a feature kind thus we use different filters to generate different feature sets.

In the convolutional layer, we implement a scalar product between each slid matrix with the filter matrix to generate the convolved matrix. ReLU (Rectified Linear Unit) is a non-linear operation and it is used after every convolution operations. Some other non-linear functions such as *sigmoid* or *tanh* can also be used but ReLU has been popularly chosen because it more computationally efficient than the others.

Pooling Layer

The output of the convolutional layer is then put into the pooling layer and processed by a pooling operator which can be of different types such as Average, Max, Sum. Among them, Max Pooling is the most popular choice, in which we define a spatial neighborhood and take the largest element from the rectified convolved matrix within that window.

It is worth to emphasize that the unit containing a Convolution and Pooling layer will form a basic block of every CNN models. In a CNN architecture with multiple blocks, the output of the right previous block will be the input for the current block.

Fully Connected Layer

The last pooling layer results in a vector of real numbers which are actually the features of the input in varied abstract levels. The Fully Connected layer aims to use these features as input to perform the classification task.

B. Representing Questions

In the task of measuring question similarity, we are given two questions which are formed in natural language sentences. Firstly we have to represent each question as a matrix of real numbers as required by CNN models. Fortunately, by using Word2Vec methods, we can represent a word as a vector, often called word vectors, where each vector element is a real number. Consequently, a question can be represented as a matrix of real numbers, where the rows stand for word vectors and the size of columns is equal the dimension of word vectors.

It is worth to emphasize that word vectors are actually vectors in a semantic space representing the meaning of words, that enables us to make computational operators on the words, such as the synonymous words by the cosine metric between word vectors. Word vectors also considered as semantic distributed representations of words, then it is usually understood as word embeddings (i.e. embed words in a semantic space).

C. A CNN Based Model for Measuring Question Similarity

In this study, we proposed a CNN-based model to estimate similarity score between the input question and related questions and then ranking related questions based on the obtained scores. The reason we use CNN in this task is that CNN can capture both long-range dependencies and features of n-gram [24]. These strengths make CNN useful for handling long questions.

Our models are illustrated in Fig. 2, which consists of four main layers: (1) an input layer for encoding words into vectors (i.e. word embeddings), (2) a convolutional layer to extract the higher-level features from the input layer, (3) a pooling layer to determine the most relevant features, and finally an MLP to determine the similarity between the input questions and related questions.

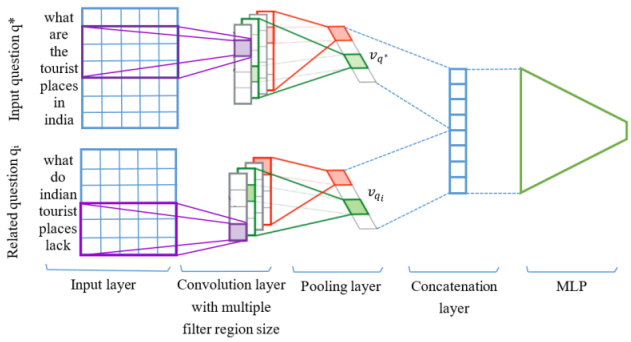

Fig. 2. The architecture of the proposed CNN-based model for computing the similarity between the input question and a related question.

Input layer

The input layer transforms each question into a word

embedding matrix, denoted by $E \in R^{a \times b}$, where a is the number of words in the question and b is the dimension of word embeddings. The obtained matrices are used as inputs for CNN's components in the proposed model.

In this layer, we firstly use the word2vec tool [3] to represent words in vector form, and then each question is represented by a matrix of real numbers. For example, suppose there is an input question $q = [w_1, w_2, ..., w_a]$, then q is represented as:

$$E = \begin{bmatrix} e_{11}, e_{12}, \dots, e_{1b} \\ e_{21}, e_{22}, \dots, e_{2b} \\ \dots \\ e_{a1}, e_{a2}, \dots, e_{ab} \end{bmatrix}_{a \times b}$$

where $[e_{i1}, e_{i2}, ..., e_{ib}]$ is the b-dimension vector representation of the word w_i in q.

Convolutional layer

The convolutional layer aims to extract higher-level features from the input matrix $E \in R^{a \times b}$. This layer works based on its filters in which we design different regional sizes for the filters to achieve different types of features. Here, we fix filters' width equal to the length of the word vector (here it is b) and set filters' height with several values, denoted by h which is considered as the number of adjacent words considered together.

Given a filter $\omega \in R^{h \times b}$, a feature s_i is generated from a filter window of words embedding $[e_i : e_{i+h-1}]$ by Eq. 1. We let $[e_i : e_{i+h-1}]$ denote the sub-matrix of E from row i to row $i+h-1$.

$$s_i = g(\omega . [e_i : e_{i+h-1}] + b_i) \quad (1)$$

where $b_i \in R$ is a bias parameter, g is a non-linear function and "." is the convolution multiplication between a sub-matrix of E and the filter ω (a sum over element-wise multiplications).

In this paper, *ReLU* is used as the active function for the convolutional layer. And then, the filters are applied to all the positions of an input matrix E to produce a feature map $s \in R^{a-h+1}$.

$$s = [s_1, s_2, \dots, s_{a-h+1}] \quad (2)$$

This process will be repeated for filters with different height h to achieve different types of features.

Note that, actually after concatenating the features of two questions, the obtained vector will be the input of a neural network to estimate the similarity of the two questions. This concatenation vector contains the features of both questions, which is then put into a neural network to combine the related features between the two questions. The neural network will suppress or increase the parameters' values which show the similarity degree between two questions.

Pooling Layer and Concatenation Layer

The pooling layer aims to abstract further the features created by the convolutional layer by aggregating scores for each filter. In this study, we applied max-pooling operation on each feature map. The meaning of max-pooling operation is to choose the highest value in each dimension of the vector to capture the most important feature. With the pooling layer, we can create a fixed-length vector from the feature map and obtain the maximum value of s as the feature corresponding to this particular filter.

$$v_{q^*} = max\{s_{q^*}\} \quad (3)$$

$$v_{q_i} = max\{s_{q_i}\} \quad (4)$$

Finally, we obtain feature vectors v_{q^*} and v_{q_i} which are then passed through a concatenation layer to create a single vector as input for the MLP layer.

$$c = concatenate\{v_{q^*}, v_{q_i}\} \quad (5)$$

MLP layer

We design our MLP layer with two hidden layers, where the output of each neural in the first hidden layer is computed as Eq. 6.

$$o_j = f\left(\sum_{i=1}^{m} w_{ij} . c_i + b_{j1}\right) \quad (6)$$

The output of each neural in the second hidden layer is computed as Eq. 7.

$$o_k = f\left(\sum_j w_{jk} . o_j + b_{k2}\right) \quad (7)$$

where $\{w_{ij}\}$ and $\{w_{jk}\}$ are the weight vectors of hidden layers and f is the activation function. In the output layer, we use a single neuron to generate similarities between questions. The *sigmoid* activation function is used in this layer to produce a probability output in the range [0,1].

$$o = \sum_k w_k . o_k + b \quad (8)$$

$$\sigma(o) = \frac{1}{1 + e^{-o}} \quad (9)$$

IV. External Knowledge

In our opinion, as a general phenomenon of deep learning models, vanilla CNNs have limitations when training data is not large enough. Therefore, to enhance the power of CNN based models in this task, we used External Knowledge (EK) extracted from other aspects of exploiting question similarities, including information from the answers of the related questions, question types, and question categorization.

A. Conventional Features

Convention methods usually used features extracted from linguistic forms, which will be utilized in our model. These features include bag-of-words and some rich linguistic features such as nouns, name entities. It also includes the ratio between the number of words of the input question and a related question, and between the input question with answers of the related question. All these features are denoted by F_1.

B. Question Type

From our observations, every question usually contains quesion words such as "*who*", "*when*", "*how*", "*why*", "*which*", "*where*", or "*what*". These words help to classify

[3] https://code.google.com/p/word2vec

question types which are also very useful evidence for determining whether two questions similar or not.

To construct the question type feature for each question, in this study, we represent question words as one-hot vectors of the vocabulary V = {"*what*", "*who*", "*when*", "*why*", "*where*", "*which*", "*how*"}. For example, the question with question word "who" will be represented as the one-hot vector [0, 1, 0, 0, 0, 0, 0]. This feature vector is denoted by F_2.

C. Word Embedding

In this paper, we use the continuous Skip-gram model of the word2vec toolkit to get vector representation of words. We follow the steps below to build question vectors and answer vectors.

Each answer or question with length n is represented by a word vector $(w_1, w_2, ..., w_n)$, where w_i is word vector representation of the i^{th} word. Suppose that we need to calculate the similarity between the input question q^* and the answer a_i, where $q^* = (w_1, w_2, ..., w_n)$ and $a_i = (v_1, v_2, ..., v_m)$

For each w_i in q, we find the most similarity word vector v_j in a_i according to *Cosine* measurement as in the Eq. 10:

$$\text{score}(w_i) = \max_{1 \le j \le m} \left(cosine\_simi(w_i, v_j) \right) \quad (10)$$

where: m is the number of words in answer a_i; w_i is the word vector representation of i^{th} word in q^*; v_j is the word vector representation of j^{th} word in a_i; $cosin\_simi(w_i, v_j)$ is the Cosine similarity of two vector representations of the i^{th} word in q^* with the j^{th} word in a_i. Finally, the similarity score between the input question q^* and answer a_i is calculated by the Eq. 11:

$$\text{similarity}(q^*, a_i) = \frac{\sum_{k=1}^{n} \text{score}(w_i)}{n} \quad (11)$$

where n is the number of words in question q^*.

Do the same as above, we also calculate the similarity between the input question and a related question. As the result, based on word embedding, we have obtained new features representing the similarity between the input question and a related question and its answers. This set of obtained features is denoted by F_3.

D. Question Category

We will assign questions with corresponding category labels and then determine how two questions are similar according to their categories. Note that the question-answer pairs extracted from cQA forums (we denote this data of questions and answers by Q dataset) have been labeled with predefined set of category labels. Therefore to utilize this feature we will build a question categorization module to label the input questions.

To achieve the question category feature, we implement the following steps:

1. Determine the question category for each input question.

2. Represent each question category in a vector form and calculate the similarity between the two obtained vectors.

The question categorization module: The question categorization module aims at assigning labels to each input question q^*. To this end, we implement the following steps:

- Build a training dataset which contains the questions in the Q dataset, each question in Q has been assigned a category label (question category).
- The questions are represented as feature vectors.
- The SVM machine-learning method is used to learn the classifier.
- For each input question, we first represent it by a feature vector and use the classifier obtained at the third step to predict the category label.

The similarity measure module: This module aims to calculate the similarity between the input question and the related question according to their category labels. To this end, we implement the following steps:

- We firstly get word vectors for all questions in the Q dataset. We then generate the representative vector for each category label by compute the average of all word vectors of all questions which belong to this category label. Finally, for each category label we have a representative vector.
- The similarity score between the related question with category label A and the input question category with category label B is calculated by the Eq. 12.

$$cosin\_sim(u, v) = \frac{\sum_{i=1}^{m} u_i * v_i}{\sqrt{\sum_{i=1}^{m} (u_i)^2} * \sqrt{\sum_{i=1}^{m} (v_i)^2}} \quad (12)$$

where u and v are two m-dimensional representative vectors of category labels A and B respectively. This similarity score is denoted as the feature F_4.

In summary, combine all the features obtained above, we finally have the overall feature vector $r = \{F_1, F_2, F_3, F_4\}$ which is considered as the external knowledge source for adding to the CNN model.

V. The Extended CNN Model

In this section, we present the external version of the model in Fig. 2 in which we combine it with the external knowledge source presented in section IV. Fig. 3 illustrates our proposed model's architecture.

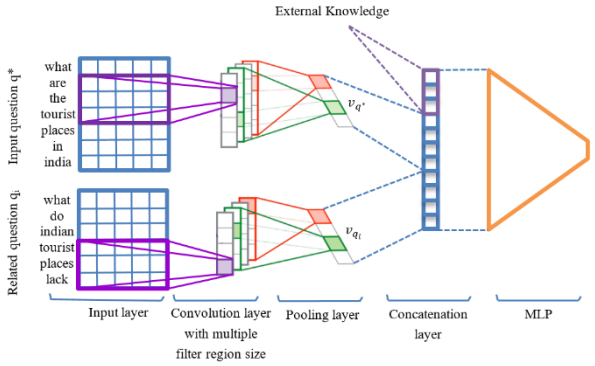

Fig. 3. The proposed CNN based model with external features for computing the similarity between two questions.

We extended the CNN-based model in Fig. 2 by identifying the concatenation layer as shown in Fig. 3, in which we combines the feature vectors created after the pooling layer of the two CNNs with the external knowledge features (i.e. the feature vector r, it was extracted as shown in Section IV). We obtain the new combined feature vector called cv computed as in Eq. 13:

$$cv = concatenate(c, r) \quad (13)$$

This combined vector is then used as input for the MLP layer of the model. The Eq. 6 is now replaced by Eq. 14:

$$o_j = f\left(\sum_{i=1}^{m+k} w_{ij}.cv_i + b_{j1}\right) \quad (14)$$

where k is the dimension of the feature vector r.

VI. Experiments

A. Dataset

In this work, we conduct experiments on the dataset SemEval-2016 task 3, subtask B[4]. This is a dataset of questions and answers extracted from cQA Qatar Living (http://www.qatarliving.com/forum). The dataset includes 337 input questions and 3369 related questions. The dataset is pre-split into 267 input questions and 2669 related questions for training, and the remain includes 70 input questions and 700 related questions for the test. Each data point is a pair of questions (an input question and the related question) and a similarity label, which is either "Relevant" or "Irrelevant".

In addition, this dataset also provides labels expressing ranks of related questions (it means a list of related questions that have been ranked according to their relevance degrees to the input question). Some statistics of this dataset are shown in Table I.

In our work we do both tasks: one is the classification which predicts a question is "Relevant" or "Irrelevant" to the input question; the other task is to predict and then rank the obtained related questions according to their similarities with the input question.

We used classification and ranking metrics to evaluate our models. The classification evaluation measurements include *Accuracy* (*Acc*), *Precision* (*P*), *Recall* (*R*), and F_1–*measure* (F_1). The ranking measures include *Mean Average Precision* (*MAP*), *Average Recall* (*AvgRec*) and *Mean Reciprocal Rank* (*MRR*).

TABLE I: The Statistics of SemEval 2016 Dataset

	Train	Test	Total
Input questions	267	70	337
Related question-answer pairs	2669-26690	700-7000	3369-33690

B. Setup Model's Experimental Configures

The models in this paper are implemented with Theano[5] from scratch. We use the accuracy on the validation set to locate the best epoch and best hyper-parameter settings for testing.

The word embeddings are re-trained using the Gensim word2vec tool[6]. The training data for the word embeddings is the dataset provided by SemEval-2016. The parameters are set as follows: (1) the word vector size is 200; (2) the maximum distance between the current word and the predicted word in a sentence is set to 5; (3) ignore all words having frequency less than 5. For comparison, we also use the 300-dimensional word vector pre-trained and provided by Google[7].

We train our models in mini-batches (the batch size is 64), and the maximum length of input questions and related questions is 256. Any tokens out of this range will be discarded. The hyper-parameters of the proposed model are summarized in Table II.

TABLE II: Hyper-Parameters of the CNN-Based Model

Descriptions	Values
word embedding	*d*=300, *d*=200
filter region size	2, 3, 4
the number of filters for each region size	128
activation function	*ReLU*
pooling	max pooling
dropout rate	0.2
batch size	64

C. Results

In this experiment we implement two models: the first one is the basic CNN-based model (in Fig. 2) which we call the "CNN-based"; the second one is the extended CNN model, i.g. the CNN combined with external knowledge features (in Fig. 3) which we call the "CNN-based + EK", where EK means external knowledge. We let both models try with two kinds of word embeddings, one with dimension of 300 and the other with dimension of 200.

The experimental results on SemEval 2016 dataset are summarized in Table 3. The first two rows show results of the CNN-based model and the next two rows are the results for the CNN-based+EK model.

The experimental results show that there is no significant improvement in the use of different dimensions of word embedding. For the basic CNN model, the classification (*Acc*) and ranking (*MAP*) measures increased by 0.14% and 0.26%, respectively. For the extended model, the classification (*Acc*) and ranking (*MAP*) measures increased by 0.29% and 0.01%, respectively. However, in both models, the use of word embeddings re-trained (*d*=200) show higher results comparing with the pre-train word embeddings (*d*=300).

From Table 3 it is clear that by using additional knowledge features the CNN-based+EK model gives much better results than the CNN-based model. The classification (*Acc*) and ranking (*MAP*) measures increased by 8.86% and 5.17%, respectively.

To compare the performance of the proposed models with the best previous studies we also make a summary as shown in Table 4. Note that all these studies conducted experiments on the same dataset (the SemEval 2016 task 3, subtask B).

The table of comparison results shows that our proposed models achieve greater efficiency with the *Accuracy* and *MAP* measures of 82.86% and 78.38% respectively.

[4] http://alt.qcri.org/semeval2016/task3/index.php?id=dataand-tools
[5] http://deeplearning.net/software/theano/
[6] www.radimrehurek.com/gensim
[7] https://code.google.com/p/word2vec

TABLE III: OUR EXPERIMENT RESULTS USING SEMEVAL 2016 DATASET

Models	Word Embedding	Classification measures				Ranking measures		
		Acc	*P*	*R*	F_1	*MAP*	*AvgRec*	*MRR*
CNN-based	*d*=300	73.71	53.65	62.19	57.60	72.95	87.87	78.29
CNN-based	*d*=200	74.00	53.65	62.81	57.87	73.21	88.35	79.24
CNN-based + EK	*d*=300	82.57	71.24	75.11	73.13	78.37	91.97	86.23
CNN-based + EK	*d*=200	82.86	72.10	75.34	73.68	78.38	92.01	86.23

TABLE IV: COMPARISON WITH PREVIOUS STUDIES ON THE SEMEVAL 2016 DATASET

Models	Classification measures				Ranking measures		
	Acc	*P*	*R*	*F1*	*MAP*	*AveRec*	*MRR*
UH-PRHLT-primary [18]	76.57	63.53	69.53	66.39	76.70	90.31	83.02
ConvKN-primary [3]	78.71	68.58	66.52	67.54	76.02	90.70	84.64
Kelp-primary [4]	79.43	66.79	75.97	71.08	75.83	91.02	82.71
SLS-primary[27]	79.43	76.33	55.36	64.18	75.55	90.65	84.64
Our (CNN-based + EK, *d*=300)	82.57	71.24	75.11	73.13	78.37	91.97	86.23
Our (CNN-based + EK, *d*=200)	82.86	72.10	75.34	73.68	78.38	92.01	86.23

VII. CONCLUSION

In this paper, we have presented our proposals for the problem identifying similar questions and ranking these questions according to their similarities to each input question in cQA systems. We have built models based on CNN to represent the questions as well as to calculate the similarities between them. Moreover, we proposed an extended model, which combines external knowledge sources with the CNN-based model which achieves the best results in comparison with the previous recent studies. The experimental results on the SemEval dataset achieved a classification accuracy (acc) of 82.86% and a ranking measure (MAP) of 78.38%.

CONFLICT OF INTEREST

The authors declare no conflict of interest.

AUTHOR CONTRIBUTIONS

The first author is responsible for organizing the steps of experimentation and preparing the manuscript. The second author is responsible for designing a framework study that helps to edit the manuscript and confirm the research steps. The last author helps to confirm the experimental results and discuss future research trends.

REFERENCES

[1] D. Buscaldi, J. L. Roux, and A. Popescu, "LIPN-CORE: Semantic text similarity using n-grams, WordNet, syntactic analysis, ESA and information retrieval based features," in *Proc. the Main Conference and the Shared Task*, 2013, pp. 162-168.

[2] G. Kondrak, *N-Gram Similarity and Distance*, Springer-Verlag Berlin Heidelberg, 2005, p. 115126.

[3] B.-C. Alberto, D. Bonadiman, and G. D. S. Martino, "ConvKN at SemEval-2016 task 3: Answer and question selection for question answering on arabic and english fora," in *Proc. SemEval-2016*, 2016, pp. 896-903.

[4] S. Filice, D. Croce *et al*., "KeLP at SemEval-2016 task 3: Learning semantic relations between questions and answers," in *Proc. SemEval-2016*, 2016, pp. 1116-1123.

[5] K. Wang, Z. Y. Ming, and T.-S. Chua, "A syntactic tree matching approach to finding similar questions in community-based qa services," in *Proc. SIGIR*, 2009, pp. 187-194.

[6] B. T. Hu, Z. D. Lu, H. Li, and Q. C. Chen, "Convolutional neural network architectures for matching natural language sentences," *Advances in Neural Information Processing Systems*, pp. 2042-2050, 2014.

[7] H. He, K. Gimpe, and J. Lin, *Multi-Perspective Sentence Similarity Modeling with Convolutional Neural Networks*, pp. 1576-1586, 2015.

[8] J. Jeon, W. B. Croft, and J. H. Lee, "Finding similar questions in large question and answer archives," in *Proc. the 14th ACM International Conference on Information and Knowledge Management*, 2005, pp. 84-90.

[9] G. Zhou, L. Cai, J. Zhao, and K. Liu, "Phrasebased translation model for question retrieval in community question answer archives," in *Proc. the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, 2011, vol. 1, pp. 653-662.

[10] X. Xue, J. Jeon, and W. Croft, "Retrieval models for question and answer archives," in *Proc. the International ACM Conference on Research and Development in Information Retrieval (SIGIR)*, 2008, pp. 475-482.

[11] X. Cao, G. Cong, B. Cui, C. S. Jensen, and C. Zhang, "The use of categorization information in language models for question retrieval," in *Proc. CIKM*, 2009, pp. 265-274.

[12] H. Duan, Y. Cao, C. Y. Lin, and Y. Yu, "Searching questions by identifying question topic and question focus," in *Proc. ACL*, 2008, pp. 156-164.

[13] Y. Cao, H. Duan, C. Y. Lin, Y. Yu, and H. W. Hon, "Recommending questions using the Mdl-based tree cut model," in *Proc. the 17th International Conference on World Wide Web*, 2008, pp. 81-90.

[14] Z. Ji, F. Xu, B. Wang, and B. He, "Question answer topic model for question retrieval in community question answering," in *Proc. the 21st ACM International Conference on Information and Knowledge Management*, 2012, pp. 2471-2474.

[15] K. Zhang, W. Wu, H. Wu, Z. Li, and M. Zhou, "Question retrieval with high quality answers in community question answering," in *Proc. the 23rd ACM International Conference on Conference on Information and Knowledge Management*, 2014, pp. 371-380.

[16] S. Filice, G. Castellucci, D. Croce, G. D. S. Martino, A. Moschitti, and R. Basili, "KeLP, a Kernel-based Learning Platform for Natural Language Processing," in *Proc. ACLIJCNLP*, 2015, pp. 19-24.

[17] P. Nakov, L. Marquez, A. Moschitti, W. Magdy, H. Mubarak, A. A. Freihat, J. Glass, and B. Randeree, "SemEval-2016 task 3: Community question answering," in *Proc. SemEval*, San Diego, California, June 2016.

[18] M. F. Salvador, S. Kar, T. Solorio, and P. Rosso, "UH-PRHLT at SemEval-2016 task 3: Combining lexical and semantic-based features for community question answering," in *Proc. SemEval*, 2016, pp. 814-821.

[19] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436-444, 2015.

[20] N. Kalchbrenner, E. Grefenstette, and P. Blunsom, "A convolutional neural network for modelling sentences," in *Proc. ACL*, 2014, pp. 655-665.

[21] Y. Goldberg, "A primer on neural network models for natural language processing," arXiv preprint arXiv: 1510.00726, 2015.

[22] M. Tan, B. Xiang, and B. Zhou, "Lstm-based deep learning models for non-factoid answer selection," arXiv: 1511.04108 [cs.CL], 2015.

[23] M. Feng, B. Xiang, M. R. Glass, L. Wang, and B. Zhou, "Applying deep learning to answer selection: A study and an open task," *CoRR*, abs/1508.01585, 2015.

[24] Y. Graves, "Supervised sequence labelling with recurrent neural networks," *SCI*, vol. 385, Springer, Heidelberg, 2012.
[25] D. S. Cicero, B. Luciano, D. Bogdanova, and B. Zadrozny, "Learning hybrid representations to retrieve semantically equivalent questions," in *Proc. ACL*, 2015, pp. 694-699.
[26] Y. Lei, K.M. Hermann, P. Blunsom, and S. Pulman, "Deep learning for answer sentence selection," *CoRR*, abs/1412.1632, 2014.
[27] M. Mitra, Y. Belinkov *et al.*, "SLS at SemEval-2016 task 3: Neural-based approaches for ranking in community question answering," in *Proc. SemEval2016*, 2016, pp. 828-835.
[28] D. Hoogeveen, Y. Li, H. Liang, B. Salehi, L. Duong, and T. Baldwin, "UniMelb at SemEval-2016 task 3: Identifying similar questions by combining a CNN with string similarity measures," in *Proc. SemEval*, 2016, pp. 851-856.
[29] J. Gehring, M. Auli, D. Grangier, and Y. N. Dauphin, "A convolutional encoder model for neural machine translation," *CoRR*, 2016.
[30] Y. Zhang and B. C. Wallace, "A sensitivity analysis of (and practitioners guide to) convolutional neural networks for sentence classification," arXiv: 1510.03820v4 [cs.CL], 2016.
[31] W.-T. Yih and X. D. He, "Christopher meek. Semantic parsing for single-relation question answering," in *Proc. ACL*, 2014, pp. 643-648.
[32] Y. L. Shen, X. D. He, J. F. Gao, and L. Deng, "Gregoire mesnil. learning semantic representations using convolutional neural networks for web search," in *Proc. the 23rd International Conference on World Wide Web*, 2014, pp. 373-374.

Van-Tu Nguyen received his bachelor's degree in information technology from Thai Nguyen University of Education in 2005, and received his master's degree in computer science from Hanoi National University of Education in 2009. He is currently a PhD student at the University of Engineering and Technology – Vietnam National University, Hanoi. He is now also a lecturer at the Faculty of Natural Science and Technology of Northwestern University in Vietnam. His research interests include machine learning and natural language processing.

Anh-Cuong Le received his bachelor's and master's degrees in information technology from the University of Engineering and Technology – Vietnam National University, Hanoi in 1998 and 2001 respectively. He received his PhD degree in information science from the Japan Advanced Institute of Science and Technology in 2007. Currently, Le Anh Cuong is an associate professor at the Faculty of Information Technology, Ton Duc Thang University in Ho Chi Minh city, Vietnam. His research interests include natural language processing, text mining and machine learning.

Ha-Nam Nguyen received the bachelor's degree in information technology at the University of Science, Vietnam National University, Hanoi in 2001, received the master's degree from Chungwoon University, South Korea in 2003 and the PhD at the University of Aviation, Korea in 2007. Currently, Nguyen Ha Nam is an associate professor at the Faculty of Information Technology, University of Engineering and Technology, Vietnam National University, Hanoi. His research areas include data mining, machine learning, data warehouse and OLAP.