Related or Duplicate: Distinguishing Similar CQA Questions via Convolutional Neural Networks

Wei Emma Zhang
Department of Computing
Macquarie University, Sydney, Australia
w.zhang@mq.edu.au

Quan Z. Sheng
Department of Computing
Macquarie University, Sydney, Australia
michael.sheng@mq.edu.au

Zhejun Tang
School of Physics and Electronic-Electrical Engineering
Ningxia University, Yinchuan, China
tangzj@nxu.edu.cn

Wenjie Ruan
Department of Computer Science
Oxford University, Oxford, UK
wenjie.ruan@cs.ox.ac.uk

ABSTRACT

Plenty of research attempts target the automatic duplicate detection in Community Question Answering (CQA) systems and frame the task as a supervised learning problem on the question pairs. However, these methods rely on handcrafted features, leading to the difficulty of distinguishing related and duplicate questions as they are often textually similar. To tackle this issue, we propose to leverage neural network architecture to extract "deep" features to identify whether a question pair is duplicate or related. In particular, we construct question correlation matrices, which capture the word-wise similarities between questions. The constructed matrices are input to our proposed convolutional neural network (CNN), in which the convolutional operation moves through the two dimensions of the matrices. Empirical studies on a range of real-world CQA datasets confirm the effectiveness of our proposed correlation matrices and the CNN. Our method outperforms the state-of-the-art methods and achieves better classification performance.

CCS CONCEPTS

• **Information systems** → **Data mining**; **Question answering**; **Specialized information retrieval**;

KEYWORDS

Question Answering; Search Quality; Convolutional Neural Networks

ACM Reference Format:
Wei Emma Zhang, Quan Z. Sheng, Zhejun Tang, and Wenjie Ruan. 2018. Related or Duplicate: Distinguishing Similar CQA Questions via Convolutional Neural Networks. In *SIGIR '18: The 41st International ACM SIGIR Conference on Research and Development in Information Retrieval, July 8–12, 2018, Ann Arbor, MI, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3209978.3210110

SIGIR '18, July 8–12, 2018, Ann Arbor, MI, USA

ACM ISBN 978-1-4503-5657-2/18/07…$15.00
https://doi.org/10.1145/3209978.3210110

1 INTRODUCTION

Community Question Answering (CQA) systems such as Stack Exchange[1] gain increasing popularity because it serves as knowledge corpus that can provide answers for subjective and open-ended questions with expert opinions. Moreover, to cater the multitude of interests for its community, Stack Exchange holds a set of subforums that focus on a particular topic or theme e.g. Language ("English"), Life ("Travel" and "Movie") and Sports ("Bicycles"). However, such crowd-sourced knowledge normally leads to varying quality of the contents [3] which may cause confusion of users.

Duplicate questions that were previously created and answered, occur frequently, despite detailed posting guidelines of Stack Exchange. Stack Exchange provides a mechanism to manually identify these duplicate questions. However, many questions remain unmarked for long time [1]. Several works target automatically detecting duplicate questions in CQA systems [1, 9, 11]. They frame the duplicate detection as a supervised learning problem and use large number of external tools to obtain features from syntax, lexicons, topic distributions, language models and knowledge bases. These handcrafted features still yield some cases that can not be correctly identified. On the one hand, related questions that share similar vocabulary are easily to be identified as duplicate. For example, "What is the correct way to pluralize an acronym?" is a related question to "What is the correct way to define an acronym when its first appearance is plural?" but identified as duplicate. On the other hand, duplicate questions that are syntactically and lexically different could be identified as non-duplicate.

In this paper, we address the first case to distinguish the duplicate and related questions. As the hand-crafted features used in existing works cannot effectively capture features from textually-similar questions, it is necessary to capture "deep" features for the question pairs. Deep learning techniques are widely adopted in the research of question answering in recent years. Covolutional neural networks (CNN) is the most commonly-used neural network architecture [2, 7, 8]. In these works, a question (or a document) is modeled as a matrix by concatenating the embeddings of words in the question. Thus one dimension of the matrix is the same as the dimension of word embeddings and the other dimension of matrix is corresponding to the length (i.e., number of words) of the question. Then a convolutional layer moves along the dimension of

[1] http://stackexchange.com/

the question, producing a vector as output. Since the values of word embedding dimensions are independent, it is meaningless to move along this dimension. However, this operation "wastes" the ability of convolutional layer to capture patterns on two dimensions.

To address this issue, we firstly propose to construct matrix with meaningful values on both dimensions. The matrix can reflect the relationship between questions in a question pair. As we model the duplicate/related distinguishing task as a binary classification problem, learning the patterns from the question relationship would lead to the classification on a question pair–they are duplicate or related. In particular, we first learn word embeddings for words in the questions and then scan through each pair of words (w_m, w_t) in a question pair (q_m, q_t) (w_m belongs to q_m and w_t belongs to q_t). For each word pair, we compute correlation between words using multiple measurements. These words correlation values construct a set of matrices for (q_m, q_t) with each measurement contributes to one matrix. We name these matrices as correlation matrices on a question pair. Then we propose a deep neural network to learn patterns from the correlation matrices. We have multiple convolutional layers as input and hidden layers. The last hidden layer is a fully-connected layer, which helps the output layer to predict whether a question pair is duplicate or related.

The rest of the paper is structured as follows. We describe proposed method in Section 2. Then we report the evaluation results in Section 3 and finally, we conclude the paper with discussion in Section 4.

2 OUR DEEP LEARNING MODEL

In this section, we introduce our convolutional neural networks for distinguishing duplicate and related questions in detail. Its main building block is the correlation matrices that model the relationship between questions. Using these correlation matrices as the input, we develop a CNN architecture that contains ten convolutional layers, a fully-connected hidden layer and an output layer with single output neuron. In the following, we first describe the correlation matrix and then give a brief explanation of the layers of our proposed architecture with their main components.

2.1 Sentence Correlation Matrices

The input of the CNN architecture is pairs of questions $\{(q_m, q_t)\}_1^N$, where N is the number of question pairs that need to be trained and examined, q_m denotes the master question and q_t represents the duplicate or related question to q_m. For each question pair, we construct a set of correlation matrices to model the relationship between the questions in the pair. As a question q is a sequence of words $[w_1, ...w_{|q|}]$ ($|q|$ denotes the number of words in q), the question can be represented as a sequence of word embeddings, which construct a matrix: $\mathbf{Q} = [\mathbf{w_1}, ...\mathbf{w}_{|q|}]$, where $\mathbf{w_i}$ is the word embedding of word w_i. Existing works use this question matrix as the input of CNN and the convolutional operation sweeps through one dimension of the matrix [2, 7, 8]. Therefore, the output of the convolutional operation is a vector representation of the question. However, the ability of convolutional operation that can extract patterns in two dimensions along a matrix is ignored. Moreover, this method cannot capture the relationship between questions by using convolutional operation.

To tackle this issue, we propose to model the relationships between questions as a matrix, named *correlation matrix*. Thus we can leverage convolutional operation to learn patterns from correlation matrix. Inspired by the work [10] that modeled the documents relationship using neural tensor network, we construct a set of correlation matrices for each pair of questions and then input these matrices into the following CNN architecture. In this way, the set of matrices can be regarded as a *tensor*, obtained by the following operation:

$$\mathbf{S} = \begin{bmatrix} \mathbf{s}_1 \\ \vdots \\ \mathbf{s}_z \end{bmatrix} = \begin{bmatrix} \varphi_1(\mathbf{Q}_m, \mathbf{Q}_t) \\ \vdots \\ \varphi_z(\mathbf{Q}_m, \mathbf{Q}_t) \end{bmatrix} \quad (1)$$

where $\mathbf{s}_i \in \mathbb{R}^{|q_m| \times |q_t|}$ is computed by correlation function φ_z and z is the number of correlation functions. Here we set z=6 and the six correlation functions adopt four commonly used vector distances and two interaction measurements. The four distances are *cosine distance*, *euclidean distance*, *correlation distance* and *mutual information*. The two interactions are adopted from text matching literatures [6] and they are *Indicator* (indicates whether the words are identical) and *Dot Product*. For example, when we adopt cosine similarity, $\varphi(\mathbf{Q}_m, \mathbf{Q}_t)$ scans through the word embeddings for each of the word in question q_m and q_t and computes the cosine similarity between word pair (w_i, w_j), where w_i belongs to words in q_m and w_j belongs to words in q_t. After finishing the scan, a correlation matrix is constructed. Taking six correlation measurements, we generate six correlation matrices for a question pair (q_m, q_t). Since the length of questions varies, we fix the matrices dimensions by setting the dimensions according to the maximum length of the questions (i.e., maximum number of words in a question). Then we add paddings in the correlation matrices for short question pairs. This operation results in the input matrices to be in $\mathbb{R}^{max(|q_m|) \times max(|q_t|)}$. Figure 1 illustrates how the correlation matrices are constructed.

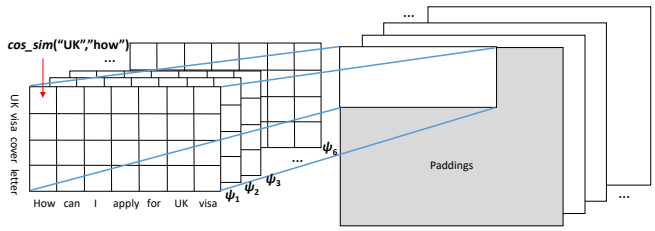

Figure 1: Constructing Question Correlation Matrices

2.2 Convolutional Layer

The goal of convolutional layer is to capture feature patterns. In our task, it will extract patterns from a set of training question pairs. The covolutional layer has several components and we explain them in this section.

Convolutional operation computes an element-wise product between a sub-matrix of an input matrix and a filter matrix, generating a smaller matrix, named *feature map*. A set of filter matrices generates a set of feature maps. For $k \times k$ filter matrices, the set of feature maps are with dimension $(max(|q_m|)\text{-}k\text{+}1) \times (max(|q_t|)\text{-}k\text{+}1)$.

The element-wise product is usually performed by a non-liner function, called activation function. We adopt *Rectified linear (ReLU)* function as it overcomes the gradient loss problem of commonly used *sigmoid* function and achieves fast convergence because of its simple form of $max(0, \mathbf{x})$.

After obtaining feature maps, we use *max-pooling* to reduces the number of neurons and corresponding parameters. *Max-pooling* only chooses the maximum value within a pool size and further reduce the size of the feature maps. The intuition is to reduce the computation complexity. Moreover, it is shown that small percentage of neurons are useful in the real learning process. Therefore, the impact on the overall performance by this operation is trivial. For $l \times l$ pools, now the feature maps are with dimension ($max(|q_m|)$-k-l+2) ×($max(|q_t|)$-k-l+2).

2.3 Hidden Layer and Output Layer

A neural network can contain several convolutional layers and each layer has convolution and max-pooling operations. Beginning from the second concolutional layer, the input of the layer is the feature maps and these layers are hidden layers.

Except for the hidden convolutional layers, the last hidden layer is a fully-connected layer. The input of this layer is the feature vectors that are obtained by flattening the feature maps from previous convolutional layers. The output of this layer is computed by a non-linear activation function $f(\mathbf{w}^T \cdot \mathbf{x} + \mathbf{b})$ that transforms the feature vectors $\mathbf{x}$ with weights $\mathbf{w}$ and bias $\mathbf{b}$ to the outputs required by the task. Typically, for multi-class classification, the output layer contains a number of neurons with each neuron holding the likelyhood of a certain class. For binary classification, the output layer has single neuron indicating positive or negative possibilities.

Our task can be modeled as a binary classification problem: duplicate pairs have positive labels while related pairs have negative labels. Therefore, we choose *sigmoid*: $f(x) = \frac{1}{1+e^{-x}}$ as the activation function in the last hidden layer. The output of *sigmoid* function is in the range of (0, 1). We then set 0.5 as the threshold to distinguish negative and positive labels.

3 EXPERIMENTS

In this section, we describe the datasets we used; the training and tunning process and the evaluation and comparison to the state-of-the-art works.

3.1 Data Preparation

Datatsets. We consider the Stack Exchange data dump in August 2017[2]. Four subsets are collected because they contain natural language question-answers and have reasonable amount of duplicate questions. Table 1 shows the statistics of the datasets. #. of Duplicates is the number of duplicate pairs in one dataset. Duplicate questions can be collected by checking "LinkTypeId" column in the data dump. #. of Related shows the number of related questions in each dataset. Related questions are crawled from the Stack Exchange web pages. They are selected and marked by Stack Exchange system and shown on the side bar of each question. As shown from the table, related pairs are much more than the duplicate question pairs. We thus down-sample the related questions to make the training and evaluation data balanced. Matrix size is the size of the correlation matrices in each dataset.

Correlation Matrices. As described in Section 2.1, to construct the correlation matrices, we first need to vectorize the words in questions. Specifically, we adopt the external pre-trained word embeddings from large corpus of text data [8]. For each dataset, we collect the word vocabulary and check the word embedding. If the pre-trained embedding does not exist, we randomly assign a value sampled from uniform distribution $U[-0.25, 0.25]$ [8]. Then we compute the similarities between each pair of words in a question pair and construct the matrix. For the four similarities, we obtain four correlation matrices for one question pair.

3.2 Model Training and Parameter Tunning

We use cross-entropy as the cost function, so the training objective is formulated as:

$$\min(-log \prod_{i=1}^{N} (y_i | \mathbf{q}_{m_i}, \mathbf{q}_{t_i}) + \lambda ||\theta||_2^2) \quad (2)$$

where y_i is the label and N is the number of training question pairs. θ contains all parameters in the proposed neural network architecture, including weights and biases in each layer. We use back propagation to compute gradients and use Adam [4], an optimization algorithm for the stochastic objective functions, to train the network. The learning rate is set to 0.0001 after several trials on 0.01, 0.005 and 0.001. Other parameters follow the settings suggested by [4]. The regularizers are used to overcome the overfitting problem especially for small and medium sized datasets. We use dropout as the regularizer and run through a small set of dropout rates (i.e., the portion of hidden units to be ignored) [0.1,0.2,0.3,0.4,0.5].

3.3 Baseline Models

We compare our model with two baseline models.

Word2Vec+Bi-CNN. In [8], Severyn et al. propose to use Word2Vec [5] word embeddings to construct question matrix. Then the question matrix is fed into a CNN, producing vector representation of the question . In parallel, the vector representation of a document is learned through another CNN. Then the authors concatenate the two vectors and train a dense neural layer. The task in [8] is to rank the documents. So the output layer outputs the similarity scores. We adapt this work for classifying the question pairs.

Doc2Vec+DNN. Inspired by [8], we directly use the Doc2Vec, an extension of Word2Vec, to vectorize the questions. Doc2Vec could learn embeddings for a sentence, paragraph sentence, paragraph or document. We learn vector representation of a question and concatenate the two vectors for a dense neural network. The intuition of this model is to compare the performance of CNN-learned

Table 1: Datasets statistics

Dataset	#. of Duplicates	#. of Related	Matrix Size
English	8,657	39,655	29×33
Travel	2,864	11,956	28×34
Movies	393	3,140	30×29
Bicycle	401	2,998	22×27

[2]https://archive.org/details/stackexchange

question vectors and Doc2Vec question vectors. We use the default parameter settings of Doc2Vec except setting the dimension of vector as 50 for the consistency to other comparing models.

3.4 Evaluation Results

We report the performance of our model and the baseline models in this section. We use F_1 score as the evaluation metric. F_1 is the harmonic mean of precision and recall, and is a measure of accuracy.

3.4.1 Performance with Parameter Tuning. We first report the performance of our model with varying learning rate and dropout rate. Table 2 shows the F_1 score on varying learning rates. Our model achieves the best performance on *Travel*, *Movie* and *Bicycle* datasets when the learning rate is 0.001. For *English* dataset, the 0.01 learning rate produces the best F_1 score. We observe from the result that inappropriate learning rates will degrade the performance of the model. Moreover, the most suitable learning rate varies among different datasets, making in the specific tuning a necessity.

Table 2: F_1 score on varying learning rates

Learning Rate	*English*	*Travel*	*Movie*	*Bicycle*
0.010	**0.673**	0.589	0.663	0.605
0.005	0.652	0.638	0.687	0.647
0.001	0.623	**0.657**	**0.694**	**0.667**

Table 3 gives the performance of our model with varying dropout rates on different datasets. For most datasets, the best F_1 score is achieved when the dropout rate is set to 0.3, i.e., 30% neuron will be dropped out. The *English* dataset is still an exception, with 0.1 as the best dropout rate. Typically, when the model is overfit, the dropout operation will help to mitigate the overfitting problem. But high dropout rate will significantly impact the performance. For the *English* dataset, we observe that higher dropout rate produces lower F_1 score. For other three datasets, this pattern is not obvious. A possible reason is that the neural network model is sensitive to the data characteristic.

Table 3: F_1 score on varying dropout rates

Dropout Rate	*English*	*Travel*	*Movie*	*Bicycle*
0.100	**0.673**	0.632	0.684	0.668
0.200	0.585	0.601	0.665	0.654
0.300	0.557	**0.657**	**0.694**	**0.667**
0.400	0.485	0.578	0.610	0.596
0.500	0.465	0.629	0.589	0.613

3.4.2 Comparison to Baseline Models. We report the performance comparison of our model and the baseline models. Table 4 presents the F_1 score of the three models. Our proposed model consistently gives the best results across the four datasets, showing the effectiveness of our model. Word2Vec+Bi-CNN performs slightly worse than our model and also achieves good F_1 score. Doc2Vec+DNN gives the lowest F_1 score on all datasets, indicating Doc2Vec vectors is less capable to capture question patterns than the CNN-learned question vectors. One reason is that when we train the Doc2Vec models, we use the questions in the dataset. But the Word2Vec vectors are learned from large external corpora. Training the Doc2Vec on larger range of natural language questions will improve the performance.

Table 4: Model comparison

	English	*Travel*	*Movie*	*Bicycle*
Doc2Vec+DNN	0.652	0.509	0.333	0.583
Word2Vec+Bi-CNN	0.670	0.623	0.618	0.653
Our Model	**0.673**	**0.657**	**0.694**	**0.667**

4 CONCLUSIONS AND DISCUSSIONS

In this paper, we define the task of distinguishing related and duplicate questions as a binary classification problem. We develop a convolutional neural network to learn features from the relationship between question pairs and classify them. One of the main contributions lies in a novel solution for constructing question correlation matrices, which are the input of the proposed neural network architecture. Our preliminary evaluations on several real-world CQA datasets confirm the effectiveness of the proposed model.

Based on the evaluations, we observe that all the models do not provide very high F_1 score. Two reasons are worth of further investigation. First, we only use question titles when performing question vectorization. Incorporating answers [11] may be helpful for capturing more representative question features, and thus enjoying better algorithm performance. Second, the prepocessing step here is very simple to keep consistency with the Word2vec+Bi-CNN model for comparison. However, this step still has space to improve, such as lemmatization or stemming.

REFERENCES

[1] Muhammad Ahasanuzzaman, Muhammad Asaduzzaman, Chanchal K. Roy, and Kevin A. Schneider. 2016. Mining Duplicate Questions in Stack Overflow. In *Proc. of MSR 2016*. 402–412.

[2] Ruey-Cheng Chen, Evi Yulianti, Mark Sanderson, and W. Bruce Croft. 2017. On the Benefit of Incorporating External Features in a Neural Architecture for Answer Sentence Selection. In *Proc. of the SIGIR 2017*. 1017–1020.

[3] Denzil Correa and Ashish Sureka. 2014. Chaff from the Wheat: Characterization and Modeling of Deleted Questions on Stack Overflow. In *Proc. of WWW 2014*. 631–642.

[4] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *CoRR* abs/1412.6980 (2014).

[5] Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *Proc. of NIPS 2013*. 3111–3119.

[6] Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, Shengxian Wan, and Xueqi Cheng. 2016. Text Matching as Image Recognition. In *Proce. of the AAAI 2016*. 2793–2799.

[7] Jinfeng Rao, Hua He, and Jimmy Lin. 2017. Experiments with Convolutional Neural Network Models for Answer Selection. In *Proc. of the SIGIR 2017*. 1217–1220.

[8] Aliaksei Severyn and Alessandro Moschitti. 2015. Learning to Rank Short Text Pairs with Convolutional Deep Neural Networks. In *Proc. of the SIGIR 2015*. 373–382.

[9] Anna Shtok, Gideon Dror, Yoelle Maarek, and Idan Szpektor. 2012. Learning from the Past: Answering New Questions with Past Answers. In *Proc. of WWW 2012*. 759–768.

[10] Long Xia, Jun Xu, Yanyan Lan, Jiafeng Guo, and Xueqi Cheng. 2016. Modeling Document Novelty with Neural Tensor Network for Search Result Diversification. In *Proc. of the SIGIR 2016*. 395–404.

[11] Wei Emma Zhang, Quan Z. Sheng, Jey Han Lau, and Ermyas Abebe. 2017. Detecting Duplicate Posts in Programming QA Communities via Latent Semantics and Association Rules. In *Proc. of WWW 2017*. 1221–1229.