Short Text Matching Method Based on Self-Learning Nearest Neighbor Graph Strategy Postprint

Authors: Cong Fu, Li Liuwu, Yang Zhenguo, Liu Wenyin

Date: 2019-05-10T00:00:00+00:00

Abstract

For the text matching problem in natural language processing, we propose a deep learning model based on a self-learning text neighbor graph framework to address short text matching. Text neighbor graphs can be obtained by using word embeddings to convert text into vector form and then constructing a text similarity relationship matrix, which can express the neighbor relationships among text samples. Existing methods typically construct static neighbor graphs, which on the one hand rely on prior knowledge, and on the other hand, make it difficult to obtain optimal representations of sentence pairs. Therefore, we propose utilizing a siamese convolutional neural network to learn a superior dynamically updated neighbor graph. The model achieves accuracy and F1 scores of 84.15% and 79.88% on the Quora dataset, and 74.55% and 81.63% on the MSRP dataset, respectively. Experiments demonstrate that the proposed model can effectively improve the accuracy of text recognition and matching.

Full Text

Preamble

Title: Self-adaptive Affinity Graph Learning for Short Text Matching

Authors: Fu Cong, Li Liuwu, Yang Zhenguo, Liu Wenyin
Affiliation: School of Computer Science, Guangdong University of Technology, Guangzhou 510006, China

Abstract: For text matching problems in natural language processing, this paper proposes a deep learning model based on a self-adaptive affinity graph learning framework for short text matching. The affinity graph can be converted into vector form using word embedding and then obtained by constructing a text similarity relationship matrix, which can express the neighbor relationships among

text samples. Current methods usually construct static affinity graphs, which rely on prior knowledge and are hard to obtain optimal representations of sentence pairs. Therefore, this paper proposes using Siamese CNN to learn better dynamically updated affinity graphs. The accuracy and F1 values of the model on the Quora dataset are 84.15% and 79.88%, respectively, and the accuracy and F1 values on the MSRP dataset are 74.55% and 81.63%, respectively. Experiments show that the proposed model can effectively improve the accuracy of text recognition and matching.

Keywords: text matching; self-adaptive affinity graph learning; word embedding; siamese CNN

0 Introduction

In recent years, with the rapid development of the Internet, information on the web has become increasingly accessible to people. However, due to the explosive growth of online information, text data has proliferated dramatically, making it a challenging task to search for needed information from such vast knowledge repositories. In response to this demand, Community Question Answering (CQA) systems have emerged and gradually become a popular Internet application, such as Yahoo Answers, Sina iAsk, and Baidu Zhidao. Users can submit questions to these Q&A communities and also answer questions posed by other users. Although modern information retrieval systems can basically meet users' needs for information finding, they do not directly provide useful relevant information during the retrieval process, requiring users to read through numerous documents to locate what they need. Consequently, current search engines cannot fully satisfy users' high demands for search quality. How can we find useful information from such massive information networks? How can we make information search more efficient and of higher quality? Research addressing these questions has found that detecting similarity in user questions and subsequently pushing the most relevant high-quality answers to users is an effective solution for such search problems.

As deep learning methods have become increasingly popular in natural language processing (NLP) applications, deep learning techniques have achieved more and more outstanding performance in solving NLP-related problems. For text similarity detection in Q&A systems, this paper proposes the AutoLMP (auto-learning match pyramid) model based on deep learning. This model employs a text content information graph generated through self-learning and a text similarity graph obtained through Pang' s method to jointly form an expressive two-channel affinity graph. The corresponding pixel values in these two channels both originate from the same pair of word vectors: one pixel value represents text content information learned by SCNN, while the other represents text similarity information calculated through prior knowledge. Subsequently, hierarchical CNN extracts rich semantic information and word-to-word relation-

ships from sentences, enabling the extraction of more complex and abundant matching information from word to sentence level. Finally, experiments on both similar question matching and paraphrase identification tasks demonstrate that the proposed method achieves promising results.

1.1 Natural Language Processing and Convolutional Neural Networks

With the rise and development of word embeddings, distributed feature representations, and neural network language models, deep learning methods have played an increasingly important role in the field of natural language processing. This section provides a detailed introduction to the application of CNNs in NLP. Due to parameter sharing in space, CNNs reduce the number of parameters in neural networks. Through multi-layer trained network spatial structures, CNNs not only significantly reduce the number of parameters but also improve training efficiency, avoiding problems such as difficult training due to excessive parameters and gradient diffusion in fully connected networks. Moreover, CNN models have also achieved good results in text classification. Originally invented for computer vision, these models were later proven effective for NLP and have achieved excellent results in semantic analysis, query retrieval, and sentence modeling. As is well known, CNN models were initially widely applied in the image domain, where after preprocessing, each image has the same pixel values for height and width, followed by convolution operations on the image. However, text processing differs from image processing. Since sentence lengths in text corpora are not fixed, they need to be processed into a structure similar to a two-dimensional matrix for experiments. On one hand, each sentence should be padded to the maximum sentence length; on the other hand, word embeddings are used to train word vectors from scratch directly in the neural network. Word vector training can employ methods such as FastText or word2vec, with the trained word vectors serving as weights for the embedding layer in the neural network, followed by fine-tuning. Kim applied the classic CNN architecture, applying convolution operations with filters of different lengths on the text matrix, where the length of word vectors in the text matrix is the same as the width of the filters. The extracted vectors were then processed using max pooling, and finally, the corresponding numbers from all filters were concatenated to obtain the sentence vector. Based on these characteristics and advantages of CNNs in NLP applications, this paper also uses CNNs to build the model.

1.2 Word Embedding

Word embedding is widely used in NLP, as this technology can convert words and text into vector forms that computers can process, which is the first step in text processing. Based on current developments, word vectorization representation is divided into three categories:

a) One-hot representation was a commonly used method in the past, where

each word is represented as a high-dimensional vector. The dimension of the vector represents the size of the vocabulary, and in each word vector, only one dimension has a value of 1 while all others are 0, with the current word represented by this 1. Therefore, word vectors represented by this method are very sparse. In addition, the shortcomings of this representation method include: (a) relationships between words are independent, unable to express information with the same or similar semantic relationships; (b) the number of word types in a sentence determines the vector dimension, which in many cases leads to a very large dictionary.

b) Distributed representation of words overcomes this disadvantage to some extent. This representation can map words to a relatively low-dimensional, dense vector space. After symbolizing words, vector formulas are used to calculate similarity between words. For words with similar contexts, their corresponding semantics are also similar.

c) Word2vec was proposed by Mikolov et al. in 2013. This representation method can effectively reduce the dimension of word vectors and consists of two models: one is the Skip-gram model, which predicts the context of a word by inputting the word itself; the other is the Continuous Bag-of-Words (CBOW) model, which learns word vector expressions by predicting the target word from its context (i.e., inputting context to predict the current word). This paper uses the CBOW model.

1.3 Text Similarity Calculation

The core of question similarity detection is text similarity calculation. In natural language processing, text similarity analysis is an important and challenging task. In recent years, deep learning models have achieved good results in many fields such as speech recognition and computer vision. In the NLP field, various model designs and methods based on deep learning have subsequently flourished. Huang et al. proposed a classic single semantic model DSSM (deep structured semantic models), which is a latent semantic model with a deep structure that can project queries and documents into a common low-dimensional space. The paper also used a technique called word hashing, which can not only effectively extend semantic models but also make them suitable for large-scale Web search applications. However, this model also has shortcomings, such as using parameter-free cosine similarity matching formulas for similarity matching tasks and ignoring the temporal relationships between words. Mueller et al. proposed a model architecture based on Siamese recurrent neural networks for learning text similarity, used for learning similarity metrics of variable-length character sequences. This model combines a stack of character-level bidirectional long short-term memory networks with a Siamese architecture, using information about similarity between string pairs to learn to project variable-length strings into a fixed-dimensional embedding space. However, in the task of professional title standardization, this model applies a manually annotated taxonomy, which is time-consuming and labor-intensive. Pang et al. proposed the Match Pyramid

model, which treats text matching as image recognition, migrating the idea of convolutional neural networks from image recognition to text matching. This model captures matching patterns at different levels—from words and phrases to entire sentences—through a matching matrix. The main idea is to model text matching as image recognition, treating the matching matrix as an image. Since this method constructs static affinity graphs, on one hand it relies on prior knowledge, and on the other hand it is difficult to obtain optimal representations of sentence pairs. Given this limitation, this paper adopts the SCNN deep learning model and proposes a method for generating learnable text content matrices with the help of SCNN to capture key information in text, thereby better identifying and detecting similar questions in Q&A systems.

2 AutoLMP Model

This paper proposes a new deep text matching framework called AutoLMP (auto-learning match pyramid). The main inspiration for this model comes from image recognition, constructing CNN-based text matching by mapping text into an image. The overall framework is shown in Figure 1 [Figure 1: see original paper].

The AutoLMP framework mainly consists of three parts: a) data preprocessing, which vectorizes sentence pairs to prepare data for affinity graph generation; b) the proposed affinity graph generation process, including both self-learning affinity graph generation and prior knowledge affinity graph generation; and c) text matching, including hierarchical CNN and multi-layer perceptron.

2.1 Building Affinity Graphs

A sentence pair shares many similarities with an image: a) the basic constituent element of a sentence is words, while that of an image is pixels; b) words in a sentence pair have more or less relationships, just as pixels in an image are intricately related; and c) sentences express certain clear information, while images also display a clear scene. Therefore, methods from image recognition can be applied to text matching tasks through pattern transformation. Inspired by Pang' s Match Pyramid method, text matching tasks can be accomplished using deep convolutional neural networks by constructing affinity graphs. The affinity graph is a relational matrix matching sentence pairs, such as a similarity matrix, which contains rich information between sentence pairs. Therefore, constructing affinity graphs is a key step in solving text matching with deep convolutional neural networks. There are many methods to construct text relationship matrices, such as calculating cosine values between words in sentence pairs in Match Pyramid. However, affinity graphs obtained through these methods are all calculated using prior knowledge, and the information they contain does not represent the optimal representation of sentence pairs, whereas neural networks can obtain better affinity graphs through learning. Therefore, this paper uses neural network structures to build the model. The following sections will detail the proposed AutoLMP model.

2.2 AutoLMP Model Details

a) Data Preprocessing. Before generating affinity graphs, this paper first vectorizes sentence pairs separately, using 50-dimensional vectors generated from a word2vec model GloVe pretrained on two million Twitter messages. Then, the word vectors of each sentence are stacked vertically, with the vertical dimension taking the maximum sentence length of 200, padding shorter lengths with zeros, resulting in a 200×50 stacking graph.

b) Text Matching. As shown in Figure 2 [Figure 2: see original paper], character-level matching refers to matching between individual words in two texts, including not only identical word matches such as Comedy-Comedy, Nights-Nights, with-with, Kapil-Kapil, live-live, but also similar word matches such as watch-see. Phrase-level matching refers to matching between phrases, i.e., N-gram matching, which involves matches of n consecutive words, such as (What is the way)-(How can), live on the sets-live show. Sentence-level matching refers to matching between sentences, composed of multiple lower-level matching units. For example, the above sentence pair can be matched at word and phrase levels. When considering matching between paragraphs containing multiple sentences, the entire paragraph is treated as a long sentence.

c) SCNN Structure. As previously discussed, constructing affinity graphs is key to transforming text matching problems into image recognition problems. An affinity graph is a two-dimensional pixel matrix, where each pixel corresponds one-to-one with words in the sentence pair, essentially mapping ordered word information onto a structural two-dimensional matrix. To solve this problem, this paper uses M to represent the affinity graph. M can be generated from the learnable stacking graph constructed in this paper through SCNN, which can be expressed by equation (1), where g and g represent stacking graphs generated from the sentence pair respectively; w_i and w_j represent sentence pair vectors learned by SCNN from g and g ; and the affinity graph M_ij is obtained by dot product of w_i and w_j. More details will be presented below.

The SCNN constructed in this paper is shown in Figure 3 [Figure 3: see original paper]. It consists of three fully convolutional neural network layers, with each convolutional layer output normalized and activated using the ReLU activation function. SCNN shares parameters and processes one sentence pair at a time, enabling sentence pairs to use the same function mapping rules, which ensures consistency in the resulting sentences.

d) SCNN Internal Parameter Settings. As shown in Figure 4 [Figure 4: see original paper], the convolution kernel sizes of the three convolutional layers are 1×5, 1×5, and 1×2, with strides of 5, 5, and 1, and kernel numbers of 16, 16, and 1 respectively. A 200×50 stacking graph yields a 200×1 vector after passing through SCNN. The significance of this convolutional layer configuration is: (a) since each row of the stacking graph represents

a word vector, the 1×N convolution kernel only learns information within the word vector without introducing noise from adjacent word vectors, as shown in Figure 4 where word vector w_i still corresponds to word vector w_j after SCNN processing; (b) the stride settings ensure that each layer learns different information, reducing redundant information. Additionally, appropriate convolution kernels guarantee sufficient model learning capacity.

e) Self-learning Affinity Graph Generation. As shown in Figure 5 [Figure 5: see original paper], the stacking graph representing each sentence pair is learned by SCNN to obtain its corresponding vector, and then one of these vectors is transposed and dot-multiplied with the other to obtain the affinity graph of the sentence pair. As shown in Figure 5, word vectors w_i and w_j are mapped to position M_ij in the affinity graph after SCNN processing. Each pixel on affinity graph M_ij represents the content information of every two words in the sentence pair, meaning that the affinity graph learned by the SCNN constructed in this paper represents the complete content information of the sentence pair.

f) Prior Knowledge Affinity Graph Generation. This paper actively adopts Pang' s method. Since the best method in their affinity graph generation is the vector dot product method, this paper also uses this method to obtain another affinity graph. The text content information graph generated through self-learning and the text similarity graph obtained through Pang' s method jointly form an expressive two-channel affinity graph. More importantly, the corresponding pixel values in these two channels both originate from the same pair of word vectors, with one pixel value representing learned text content information and the other representing text similarity information calculated through prior knowledge. This is the essence of the AutoLMP model proposed in this paper.

2.3 Key Method of AutoLMP–Hierarchical CNN

This section provides a detailed analysis of the hierarchical CNN method for text matching used in this paper. This method can extract matching patterns at different levels, including words, phrases, and sentences. For the first layer of hierarchical CNN, the k-th convolution kernel w_k slides over the affinity graph to perform convolution calculations to generate a feature map m_k, where w_k represents the size of the k-th convolution kernel. This paper uses n×n convolution kernels and ReLU activation functions. The ReLU formula is f(x) = max(0, x). Additionally, due to varying sentence lengths, this paper adopts a dynamic pooling strategy to solve this problem. The feature map obtained using dynamic pooling can be expressed as m' $k = max\{p,q\}(m\_k)$, where p and q represent the length and width of the pooling kernel corresponding to dynamic pooling. The length and width of the pooling kernel are determined by sentence pair lengths x and y. After pooling, the resulting feature map sizes are x' ×y' . After the first convolutional and dynamic pooling layers, higher-level

feature maps are obtained. When l > 2, hierarchical CNN continues to deepen layers, with subsequent convolution and max pooling generalized as m^l_k = f(w^l_k * m^{l-1} + b^l_k), where c_l represents the number of feature maps in layer l.

2.4 Analysis of Hierarchical CNN Effectiveness

CNNs can effectively extract basic visual elements such as edges and corners in image recognition. In the AutoLMP model proposed in this paper, hierarchical CNN can also extract rich semantic information and relationships between words. As shown in Figure 6 [Figure 6: see original paper], this paper will demonstrate how hierarchical CNN performs feature extraction through examples.

a) On the two-channel affinity graph, cell M_ij represents the mapping result between the i-th word of sentence 1 and the j-th word of sentence 2. As shown in Figure 6, two 3×3 convolution kernels in the first convolutional layer map the pairwise relationships of three adjacent words in the sentence pair to higher-level feature maps sequentially. This pattern is similar to how convolution kernels extract edge features in image recognition.

b) The multiple feature maps obtained from the first layer are further processed by the next convolutional layer to extract higher-dimensional features. For example, this layer still uses 3×3 convolution kernels to extract features from the feature maps. Since each cell in these feature maps represents the mapping of pairwise relationships between three adjacent words in the sentence pair, and now the model processes relationship information of nine adjacent words in the sentence pair each time, more sentence and word information is extracted, similar to how deeper networks in image recognition extract more abstract image features.

From the above analysis, it can be seen that hierarchical CNN can extract more complex and abundant matching information from word to sentence level.

2.5 Prediction Network and Objective Function

In the AutoLMP model, this paper uses a multi-layer perceptron to predict matching scores and then obtains a binary classification result from the multi-layer perceptron. Using a two-layer perceptron as an example, the calculation of the perception layer is as follows: s = (W_2(W_1m + b_1) + b_2), where s is the matching probability of the sentence pair; m is the output of hierarchical CNN; W_i is the weight of the i-th perception layer; and represents the activation function.

In the AutoLMP model, this paper uses the Sigmoid function to process the model output to generate category prediction probabilities for sentence pairs, and uses binary cross-entropy loss as the objective function for training. The final optimization minimizes the loss: loss = -1/N Σ_{i=1}^N [l_i log(s_i) +

(1-l_i)log(1-s_i)], where l_i is the label of the i-th sentence pair; s_i is the predicted probability of the i-th sentence pair.

3 Experiments

This chapter conducts experiments on two tasks—similar question matching and paraphrase identification—to demonstrate the superiority of AutoLMP over baselines.

3.1 Datasets

1) **Quora Duplicate Question Dataset**
   As a popular knowledge-sharing platform, Quora should not have similar questions appearing multiple times. This paper adopts Quora' s publicly available question dataset, containing over 400,000 question pairs. Detecting whether a question pair is semantically similar involves taking two questions q1 and q2 as input, processing them through the model, and determining whether these two questions have a semantic similarity relationship based on the output probability value, where results closer to 1 indicate more similar question pairs. The dataset contains approximately 63% non-duplicate questions and 37% duplicate questions.

2) **MSRP Dataset**
   To further verify the generalization and effectiveness of the proposed AutoLMP model, this paper also tests on the benchmark MSRP dataset. This dataset contains 4,076 instances for training and 1,725 instances for testing, aimed at determining whether two sentences have the same meaning, belonging to the text matching task.

3.2 Method Effectiveness Demonstration

As shown in Figure 7 [Figure 7: see original paper], the two matrices are the matching matrix generated by Pang' s method and the self-learning content information matrix generated by the AutoLMP method. Comparing the two matrices reveals that the self-learning content information matrix has certain feature distributions similar to the matching matrix. As indicated by the four red boxes on the left and right, both the matching matrix and self-learning content information matrix have many white blocks, meaning that features present in the matching matrix are also present in the self-learning content information matrix, which even contains more feature information. In summary, the self-learning content information matrix generated by the proposed AutoLMP method can represent content information features of sentence pairs and forms a neighbor relationship with the matching matrix, enabling the two-channel affinity graph composed of the two matrices to express rich sentence pair information. After multiple experiments, it was found that although the feature distribution of the self-learning content information matrix varies significantly due to different sentence pairs, it generally demonstrates certain features.

3.3 Method Comparison

To verify the effectiveness of the AutoLMP model, this paper conducts experimental comparisons with multiple methods. The following models are used: first, the experimental datasets are preprocessed, including tokenization, stemming, text case conversion, and stop word removal; then the normalized datasets are trained, and AutoLMP automatically learns filter weight values based on the task requirements. For text matrix graphs, to test the effectiveness and reliability of different text matrix generation methods, this paper sets up two groups of experiments, each using different classical methods such as TF-IDF, DSSM, ARC-I, MP-DOT, etc., for comparison with the proposed AutoLMP model. The following are introductions to each method:

a) **TF-IDF**. TF-IDF is a widely used method in text mining. In this method, each text is represented as a |V|-dimensional vector, where each element represents the TF-IDF score of the corresponding word in the text, and |V| is the vocabulary size. In this paper' s experiments, IDF scores are calculated across the entire dataset, and the final matching score is produced by the inner product of the two vectors.

b) **DSSM / CDSSM**. Since DSSM and CDSSM require large amounts of data for training, this paper directly uses published models to train the test data through large datasets.

c) **ARC-I / ARC-II**. This paper uses ARC-I and ARC-II. Since no public code is available, this method uses the same implementation as the original paper.

3.4 Implementation Details

Through multiple rounds of testing, this paper finally determined the hyperparameter settings. The training process uses stochastic gradient descent (SGD) for error backpropagation, with the learning rate set to 0.1. For error updates, batch processing is adopted, with each batch containing 64 samples (batch size = 64), dropout set to 0.5, and the number of iteration epochs set to 500. The parameter set with the best performance on the evaluation dataset is selected as the final trained model parameters.

The experimental results are shown in Table 1 . It can be seen that among traditional methods, ARC-II has achieved over 80% accuracy, although these methods only use relatively low-level text feature information. The MP-DOT method, which obtains affinity graphs through prior knowledge, shows significant improvement on this test set, achieving 83.48% accuracy and 79.37% F1 value. Additionally, the Single AutoLMP method proposed in this paper (using only the single-channel affinity graph obtained through self-learning) achieves 82.77% accuracy and 78.96% F1 value, while the complete AutoLMP method proposed in this paper achieves the best performance, with accuracy reaching 84.15% and F1 value reaching 79.88%. Experimental results demonstrate that

modeling text matching tasks as image recognition is a good solution. The results further prove the superiority of the proposed AutoLMP method, which utilizes both affinity graphs obtained through prior knowledge and those obtained through learning. The two-channel affinity graph contains richer text feature information, and hierarchical CNN can effectively extract text feature information.

3.6 Paraphrase Identification

The experimental results are listed in Table 2 . It can be seen that traditional simple models such as TF-IDF have already achieved high precision of about 69.32%, although they only use single-letter matching signals. The method proposed in this paper performs better than TF-IDF, indicating that the complex matching patterns captured by hierarchical convolution are important for text matching tasks. Through comparison with recent deep models, it can be seen that the proposed model (AutoLMP) outperforms all models. Based on the hierarchical CNN method, whether using single-channel MP-DOT or single-channel Single AutoLMP, better experimental results are obtained than traditional methods. Although the two-channel AutoLMP method proposed in this paper only shows a slight improvement over the previous MP-DOT method, this proves that the method is feasible and still has significant room for improvement. In future work, this paper will conduct more in-depth research on self-learning affinity graph generation to further improve the method.

4 Conclusion

This paper proposes an effective hierarchical CNN method based on self-learning affinity graphs. After encoding text information as word vectors, learnable SCNN is used to obtain affinity graphs, which are then combined with prior knowledge to obtain Matching Matrices, resulting in a more comprehensive semantic information two-channel text matching matrix. Through multiple comparative experiments, the method can better match similar texts, proving its feasibility and effectiveness. The next step will further explore more hierarchical and granular semantic analysis tasks for text, seeking deep learning methods more suitable for text semantic similarity analysis.

References

[1] Li Xin. Research on paragraph retrieval technology for question answering system [D]. Hefei: University of Science and Technology of China, 2010.

[2] Lin Yiou, Lei Hang, Li Xiaoyu, et al. Deep learning method and application in natural language processing [J]. Journal of University of Electronic Science and Technology of China, 2017, 46 (6): 913-919.

[3] Mikolov T, Kai Chen, Corrado G, et al. Efficient estimation of word representations in vector space [J/OL]. Computer Science, 2013. arXiv:1301.3781v3

[4] Wei Xu, Rudnicky A. Can artificial neural networks learn language models? [C]// Proc of the 6th International Conference on Spoken Language Processing. Beijing: ICSLP Press, 2000.

[5] Lecun Y L, Bottou L, Bengio Y, et al. Gradient-based learning applied to document recognition [J]. Proceedings of the IEEE, 1998, 86 (11): 2278-2324.

[6] Shen Yelong, He Xiaodong, Gao Jianfeng, et al. Learning semantic representations using convolutional neural networks for Web search [C]// Proc of International Conference on World Wide Web. 2014: 373-374.

[7] Joulin A, Grave E, Bojanowski P, et al. Bag of tricks for efficient text classification [J/OL]. Preprint arXiv 2016: 1607.01759.

[8] Kim Y. Convolutional neural networks for sentence classification [C]// Proc of Conference on Empirical Methods in Natural Language Processing. 2014: 1408.

[9] Graves A, Mohamed A R, Hinton G. Speech recognition with deep recurrent neural networks [C]// Proc of IEEE International Conference on Acoustics, Speech and Signal Processing. Piscataway, NJ: IEEE Press, 2013: 6645-6649.

[10] Young T, Hazarika D, Poria S, et al. Recent trends in deep learning based natural language processing [J]. IEEE Computational Intelligence Magazine, 2017, 13 (3): 55-75.

[11] Huang Posen, He Xiaodong, Gao Jianfeng, et al. Learning deep structured semantic models for Web search using clickthrough data [C]// Proc of the 22nd ACM International Conference on Information & Knowledge Management. New York: ACM Press, 2013: 2333-2338.

[12] Mueller J, Thyagarajan A. Siamese recurrent architectures for learning sentence similarity [C]// Proc of AAAI Conference on Artificial Intelligence. 2016: 2786-2792.

[13] Liang Pang, Yanyan Lan, Jiafeng Guo, et al. Text matching as image recognition [J/OL]. CoRR, 2016: abs//1602.06359.

[14] Hu Baotian, Lu Zhengdong, Li Hang, et al. Convolutional neural network architectures for matching natural language sentences [C]//Proc of the 27th International Conference on Neural Information Processing Systems. Cambridge, MA: MIT Press, 2014: 2042-2050.

Note: Figure translations are in progress. See original paper for figures.