A Survey of State-of-the-Art Short Text Matching Algorithms

Weiwei Hu[1(✉)], Anhong Dang[1], and Ying Tan[2]

[1] State Key Laboratory of Advanced Optical Communication Systems and Networks, School of Electronics Engineering and Computer Science, Peking University, Beijing 100871, China
{weiwei.hu,ahdang}@pku.edu.cn

[2] Key Laboratory of Machine Perception (MOE), and Department of Machine Intelligence, School of Electronics Engineering and Computer Science, Peking University, Beijing 100871, China
ytan@pku.edu.cn

Abstract. The short text matching task uses an NLP model to predict the semantic relevance of two texts. It has been used in many fields such as information retrieval, question answering and dialogue systems. This paper will review several state-of-the-art neural network based text matching algorithms in recent years. We aim to provide a quick start guide to beginners on short text matching. The representation based model DSSM is first introduced, which uses a neural network model to represent texts as feature vectors, and the cosine similarity between vectors is regarded as the matching score of texts. Word interaction based models such as DRMM, MatchPyramid and BERT are then introduced, which extract semantic matching features from the similarities of word pairs in two texts to capture more detailed interaction information between texts. We analyze the applicable scenes of each algorithm based on the effectiveness and time complexity, which will help beginners to choose appropriate models for their short text matching applications.

Keywords: Short text matching · Deep learning · Representation learning · Neural networks

1 Introduction

Short text matching is a widely used NLP technology which aims to model the semantic relationship between two texts. Information retrieval, question answering and dialogue systems are the main application areas of short text matching.

In information retrieval, users want to find relevant documents for a given query. How to match the query with appropriate documents is crucial to search engines. Text matching can also be used to match question with appropriate answers, which is very helpful in automatic custom service and will significantly reduce the labor cost.

Y. Tan and Y. Shi (Eds.): DMBD 2019, CCIS 1071, pp. 211–219, 2019.
https://doi.org/10.1007/978-981-32-9563-6_22

Recent researches show that neural network based text matching algorithms outperform traditional text matching algorithms such as TFIDF, latent semantic analysis (LSA) [3] and latent Dirichlet allocation (LDA) [2]. Using neural networks to represent text and learn the interaction pattern between texts will make the model able to mine the complex semantic relationship of texts.

Many neural network based text matching algorithms have been proposed in recent years. This paper only focuses several state-of-the-art algorithms among them. Considering the effectiveness and time efficiency, DSSM, DRMM, MatchPyramid and BERT are chosen to present in this paper. There are many other famous text matching algorithms, such as ARC [9] and DURT [13]. Due to the space limit, we will not introduce them in this paper. An experimental evaluation of different text matching algorithms can be found in [5,15].

2 Deep Structured Semantic Models (DSSM)

DSSM is a well-known short text matching algorithm, which is the abbreviation for deep structured semantic models [10]. It is first proposed to match query and documents in web search applications. DSSM uses neural networks to represent queries and documents as vectors. The vector distance between a query and a document is regarded as the matching score of them. There are mainly three kinds of neural networks to represent text for DSSM, and we will introduce them in the following three sub-sections.

2.1 Feed-Forward Network Based DSSM

The basic DSSM algorithm uses feed-forward networks to represent queries and documents. The architecture of basic DSSM is shown in Fig. 1.

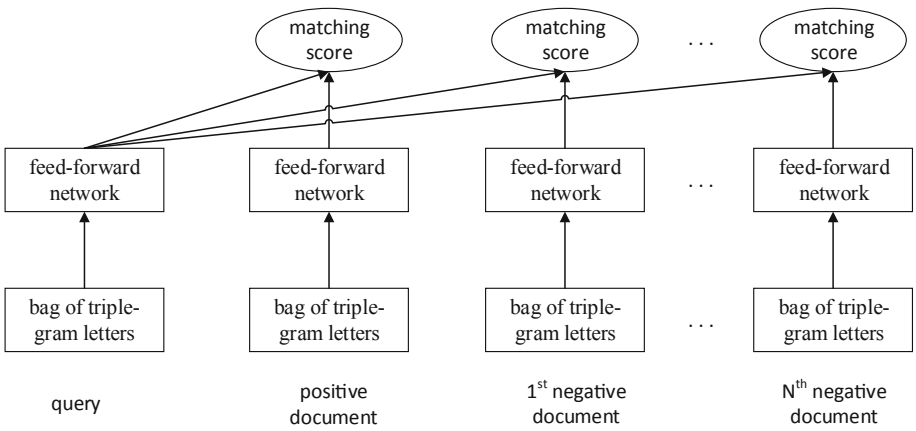

Fig. 1. The architecture of DSSM

The input query and documents are firstly represented as bag of triple-gram letters by word hashing. Word hashing by triple-gram of letters has two advantages. The first one is that it is able to significantly reduce the size of the vocabulary. A small vocabulary will make calculation of the bottom layer of neural network very fast. The second advantage is that triple-gram representation of query letters is more robust to misspelling than word based representation. The search engine usually faces a large number of users, and many users may misspell the queries. For misspelled words, the correct parts will be preserved by triple-gram letters.

The bag of triple-gram letters is then represented as a boolean vector. If a triple-gram is present in a query or a document, the corresponding dimension of the boolean vector is set to one. Otherwise, it is set to zero. This vector is fed into a feed-forward network. The network will output a vector which is the semantic representation of the query or the document.

The cosine similarity between a query and a document is regarded as the matching score of their similarity. Let the cosine similarity between a query and its positive document vector be s^+, and the cosine similarity with its i-th negative document be s_i^-, assuming there are N negative documents in total. The loss function for training DSSM is shown in Formula 1.

$$loss = -log\frac{e^{\gamma s^+}}{e^{\gamma s^+} + \sum_{i=1}^{N} e^{\gamma s_i^-}}. \tag{1}$$

γ in Formula 1 is the hyper-parameter to control the smoothness of the softmax function. Minimizing this loss function will push the matching score of positive document become larger and push the matching scores of negative documents become smaller. Such process will make DSSM able to discriminate relevant documents from irrelevant ones for a given query.

2.2 Convolutional DSSM (CDSSM)

The basic DSSM only uses feed-forward networks to represent the query and documents. However, feed-forward networks are not originally designed for processing sequential data, while query and documents are sequence of words. To better capture the sequential nature of query and document text, the DSSM with convolutional-pooling structure (abbr. CDSSM) is proposed [19].

CDSSM uses a sliding window to split the input text (i.e. a query or a document) into word n-grams. In each sliding window, the triple-gram letters are extracted for each word, and each word is represented as a boolean vector of triple-gram letters. The boolean vectors of all words in the sliding window are concatenated as the feature vector of the sliding window.

The convolutional operation is applied on each sliding window. The feature vector of each sliding window is fed into a feed-forward network, in order to extract higher level semantic feature of the sliding window. The max-pooling layer takes the element-wise maximum of the network's output vectors of all sliding windows. The variable-length input text will become a feature vector

with fixed length. At last another feed-forward network is applied on top of the max-pooling layer, and output the final semantic feature vector of the input text.

2.3 LSTM-DSSM

For many sequential applications, long short-term memory (LSTM) [8] works better than convolutional networks. LSTM-DSSM [16] is a variant of DSSM which uses LSTM to extract the feature vector of input text.

LSTM is a special kind of recurrent neural networks [6]. Recurrent neural networks process the input text sequentially, and use recurrent layers to remember previous states. For long sequences, the recurrent neural networks may encounter the problem of gradient vanishing during training. LSTM introduces input gate, output gate and forget gate to the recurrent layers. The opening and closing of the gates make LSTM able to reserve long-term information for long sequences.

3 Deep Relevance Matching Model (DRMM)

DRMM [7] matches two pieces of text in word level. Word interaction matrix is first calculated, and then the high-level feature is extracted from the word interaction matrix.

Supposing the embeddings of query words are represented as $q_1, q_2, ..., q_Q$, and the embeddings of the document words are represented as $d_1, d_2, ..., d_D$, where Q is the length of the query and D is the length of the document.

For the i-th query word q_i, a term gate is first applied to get its weight among all the query words. The term gate g_i is calculated as Formula 2.

$$g_i = \frac{e^{wq_i}}{\sum_{j=1}^{Q} e^{wq_j}}. \tag{2}$$

w in Formula 2 is a learnable parameter. It can be seen that g_i is a normalized weight of the i-th query word, and the term gates of all query words sum up to 1.

To match the document for the i-th query word, the cosine distance of q_i with all the words in the document is calculated. That is:

$$s_{ij} = cosine(q_i, d_j) \tag{3}$$

For the i-th query word, the histogram of s_{i1} to s_{iD} is calculated. The range of cosine distance is between -1 and 1. This range is split into several intervals and the number of cosine values fallen in each interval is calculated. The counts of words in all intervals forms a feature vector, which is able to reflect the matching status of the i-th query word with the document. A feed-forward network with one output neuron is applied on this feature vector. Let the output of the network be m_i, which can be regarded as the matching score of the i-th query word with the document.

The final matching score of the query and the document is calculated as follows:

$$score = \sum_{i=1}^{Q} g_i m_i. \tag{4}$$

That is, the matching score is the weighted sum of all the query words' matching scores with the document.

The original DRMM paper uses histogram of bins to represent the interaction feature. Besides, the maximum k elements of s_{i1} to s_{iD} can also be used as an interaction feature, where k is a hyper parameter which controls the feature size. Using the top k elements from the word matching scores will make the interaction between query and document become more finer-grained. Such feature usually performs better than the original DRMM with the histogram feature.

4 MatchPyramid

DRMM uses human-defined features on top of the matching matrix to calculate the matching score. Instead of using human-defined feature, MatchPyramid [17] uses convolutional neural networks to learn feature representations from the matching matrix.

The architecture of MatchPyramid is shown in Fig. 2.

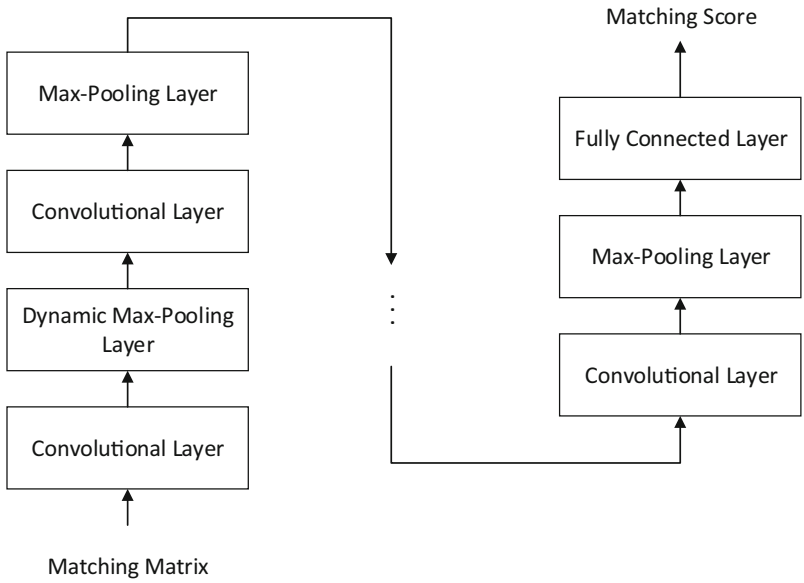

Fig. 2. The architecture of MatchPyramid.

The matching matrix between the query word sequence and the document word sequence is firstly calculated. The matching matrix is then regarded as an 2D image and convolutional neural networks is used to predict the matching score. A serial of alternating convolutional and max-pooling layers are applied on the image of matching matrix. The first max pooling layer uses dynamic pooling

size to convert text with variable length to fixed size. Full connected layer is used to output the matching score of MatchPyramid.

Due to the representation learning nature of convolutional neural networks, MatchPyramid is able to learn the hierarchical interaction between query words and document words, and mine complex matching pattern between texts.

5 Bidirectional Encoder Representations from Transformers (BERT) for Short Text Matching

BERT is a pre-trained text encoder model which is based on the transformer architecture [4]. It has achieved state-of-the-art performance on many tasks, such as GLUE benchmark [21] and SQUAD [18].

BERT uses transformer to encode a word sequence of text into an output vector sequence. Transformer is firstly proposed for machine translation [20]. The encoder of transformer consists of a serial of layers. Each layer can be further divided into two sublayers, i.e. the multi-head self-attention layer and the feed-forward layer. Due to the space limit, we will not describe the detailed structure of transformer here.

BERT firstly pre-trains the transformer encoder on unsupervised text corpus. Traditional NLP pre-training algorithms use language model to predict next words. BERT introduces two new pre-training methods. The first method is to randomly mask out some words in a sentence, and then use the output of transformer encoder to predict these masked words. The second method is to predict whether two sentences are adjacent in the corpus.

After pre-training the transformer encoder, many specific tasks can be fine-tuned on the trained model. The architecture for fine-tuning text matching task in shown in Fig. 3.

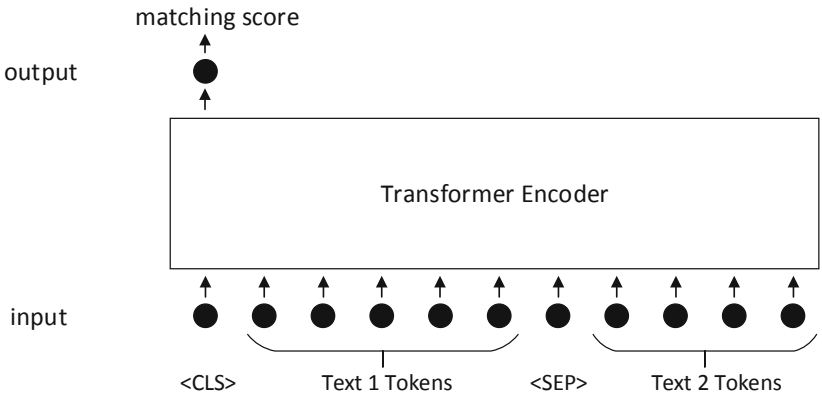

Fig. 3. BERT for text matching.

In the input layer, the two texts are concatenated as one sequence. A special $<SEP>$ token is added between the two sequences. A beginning token $<CLS>$ is inserted at the beginning of BERT input. The output corresponding to the $<CLS>$ is used to predict the matching score.

Using BERT to match short text will take full advantage the interaction feature between two texts because of the self-attention mechanism of transformer encoder. In many text matching task such question answering, BERT has outperformed existing algorithms by a large margin.

6 Model Selection in Real-World Applications

DSSM uses cosine similarity between feature vectors to measure the matching score between two texts. The calculation of cosine similarity is usually very fast. This makes the DSSM very suitable for online services which have real-time requirements on inferring the text similarity, on condition that the feature vectors have been pre-calculated offline.

For example, in search engines, the feature vector of frequent queries and all documents can be calculated in offline machines. The online servers only need to calculate infrequent queries. The matching scores of a query and candidate documents can be simply calculated by the cosine distances. This process is able to return search results quickly and will significantly reduce the load of online servers.

Another applicable scene of DSSM is when we need to find semantic similar texts from a huge collection of texts. It is inefficient to calculate matching scores between the query text and all texts in the huge collection, since the time complexity is $O(N)$ where N is the size of the huge collection. For cosine similarity, the approximate nearest neighbor (ANN) search [1,14] can be used to accelerate the process of searching for nearest neighbors. ANN uses some index structures to avoid the comparison with all samples in the huge collection, and is able to control the time of searching for similarity vectors from a huge collection in an acceptable range. State-of-the-art ANN algorithms includes HNSW [12] and Faiss [11], etc.

DSSM does not uses direct word level interaction feature to calculate the matching score, which may limit its effectiveness on matching short texts. DRMM, MatchPyramid and BERT are able to result in better matching performance, because they take advantages of fine-grained matching information based on word interaction feature. However, such word interaction based algorithms cannot be pre-calculated offline and are unable to be accelerated in online services like the way of DSSM.

If the online efficiency is crucial, DRMM is a good choice for online services. The main components of DRMM are feed-forward networks, which are rather faster than other kinds of neural networks. BERT usually performs better than other algorithms, but its architecture is too complex and its time complexity is much higher than other algorithms. It is suitable for the applications that the time efficiency is not the major constraint and the state-of-the-art matching

performance has much more positive influences. MatchPyramid can be a trade-off between the matching performance and the time efficiency.

7 Conclusion

This paper reviews some state-of-the-art short text matching algorithms. DSSM uses a feature vector to represent each text and the vector distances is regarded as the matching score. Convolutional networks and LSTM are able to further enhance the representation ability of DSSM. DRMM extracts features from word interaction matrix between words of two text, which is able to better capture the interaction information between text. MatchPyramid regards the matching matrix as an image, and uses convolutional neural networks to predict the matching score. BERT utilizes the unsupervised pre-trained model to enhance the feature learning of text. It uses the self-attention mechanism of transformer encoder to learn the interaction between two texts, which is able to reflect higher level semantic interaction.

Acknowledgments. This work was supported by National Key Research and Development Program of China under grant no. 2016QY02D0304.

References

1. Aumüller, M., Bernhardsson, E., Faithfull, A.: ANN-benchmarks: a benchmarking tool for approximate nearest neighbor algorithms. Inf. Syst. (2019)
2. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent Dirichlet allocation. J. Mach. Learn. Res. **3**(Jan), 993–1022 (2003)
3. Deerwester, S., Dumais, S.T., Furnas, G.W., Landauer, T.K., Harshman, R.: Indexing by latent semantic analysis. J. Am. Soc. Inf. Sci. **41**(6), 391–407 (1990)
4. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018)
5. Fan, Y., Pang, L., Hou, J., Guo, J., Lan, Y., Cheng, X.: MatchZoo: a toolkit for deep text matching. arXiv preprint arXiv:1707.07270 (2017)
6. Graves, A.: Supervised sequence labelling. In: Graves, A. (ed.) Supervised Sequence Labelling with Recurrent Neural Networks. SCI, vol. 385, pp. 5–13. Springer, Berlin (2012). https://doi.org/10.1007/978-3-642-24797-2_2
7. Guo, J., Fan, Y., Ai, Q., Croft, W.B.: A deep relevance matching model for ad-hoc retrieval. In: Proceedings of the 25th ACM International on Conference on Information and Knowledge Management, pp. 55–64. ACM (2016)
8. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
9. Hu, B., Lu, Z., Li, H., Chen, Q.: Convolutional neural network architectures for matching natural language sentences. In: Advances in Neural Information Processing Systems, pp. 2042–2050 (2014)
10. Huang, P.S., He, X., Gao, J., Deng, L., Acero, A., Heck, L.: Learning deep structured semantic models for web search using clickthrough data. In: Proceedings of the 22nd ACM International Conference on Information & Knowledge Management, pp. 2333–2338. ACM (2013)

11. Johnson, J., Douze, M., Jégou, H.: Billion-scale similarity search with GPUs. arXiv preprint arXiv:1702.08734 (2017)
12. Malkov, Y.A., Yashunin, D.A.: Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. IEEE Trans. Pattern Anal. Mach. Intell. (2018)
13. Mitra, B., Diaz, F., Craswell, N.: Learning to match using local and distributed representations of text for web search. In: Proceedings of the 26th International Conference on World Wide Web, pp. 1291–1299. International World Wide Web Conferences Steering Committee (2017)
14. Naidan, B., Boytsov, L., Nyberg, E.: Permutation search methods are efficient, yet faster search is possible. Proc. VLDB Endow. **8**(12), 1618–1629 (2015)
15. NTMC-Community: Matchzoo (2017). https://github.com/NTMC-Community/MatchZoo/tree/1.0
16. Palangi, H., et al.: Semantic modelling with long-short-term memory for information retrieval. arXiv preprint arXiv:1412.6629 (2014)
17. Pang, L., Lan, Y., Guo, J., Xu, J., Wan, S., Cheng, X.: Text matching as image recognition. In: Thirtieth AAAI Conference on Artificial Intelligence (2016)
18. Rajpurkar, P., Zhang, J., Lopyrev, K., Liang, P.: SQuAd: 100,000+ questions for machine comprehension of text. arXiv preprint arXiv:1606.05250 (2016)
19. Shen, Y., He, X., Gao, J., Deng, L., Mesnil, G.: A latent semantic model with convolutional-pooling structure for information retrieval. In: Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, pp. 101–110. ACM (2014)
20. Vaswani, A., et al.: Attention is all you need. In: Advances in Neural Information Processing Systems, pp. 5998–6008 (2017)
21. Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., Bowman, S.R.: GLUE: a multi-task benchmark and analysis platform for natural language understanding. arXiv preprint arXiv:1804.07461 (2018)