A Study of MatchPyramid Models on Ad-hoc Retrieval

Liang Pang Yanyan Lan Jiafeng Guo Jun Xu Xueqi Cheng
CAS Key Lab of Network Data Science and Technology,
Institute of Computing Technology, Chinese Academy of Sciences
pangliang@software.ict.ac.cn, {lanyanyan, guojiafeng, junxu, cxq}@ict.ac.cn

arXiv:1606.04648v1 [cs.IR] 15 Jun 2016

ABSTRACT

Deep neural networks have been successfully applied to many text matching tasks, such as paraphrase identification, question answering, and machine translation. Although ad-hoc retrieval can also be formalized as a text matching task, few deep models have been tested on it. In this paper, we study a state-of-the-art deep matching model, namely MatchPyramid, on the ad-hoc retrieval task. The MatchPyramid model employs a convolutional neural network over the interactions between query and document to produce the matching score. We conducted extensive experiments to study the impact of different pooling sizes, interaction functions and kernel sizes on the retrieval performance. Finally, we show that the MatchPyramid models can significantly outperform several recently introduced deep matching models on the retrieval task, but still cannot compete with the traditional retrieval models, such as BM25 and language models.

CCS Concepts

•Information systems → Retrieval models and ranking;

Keywords

Deep Matching Models, Ranking Models, Convolutional Neural Networks

1. INTRODUCTION

Many text based applications, such as paraphrase identification, question answering, and ad-hoc retrieval, can be formalized as a matching task [5]. Recently, a variety of deep neural models have been proposed to solve such kind of text matching tasks. However, most proposed deep matching models were only evaluated on the natural language processing tasks such as paraphrase identification and question answering [10, 7]. Few deep model has been tested on the ad-hoc retrieval task. Even the models proposed for the Web search tasks, including DSSM [3] and CDSSM [9], were only tested on the <query, doc title> pairs which are not a typical ad-hoc retrieval setting.

In this paper, we propose to study the performance of deep matching models on the ad-hoc retrieval task. We choose a recently introduced deep matching model, namely MatchPyramid [6], which has been shown state-of-the-art performances on several text matching tasks. In MatchPyramid, local interactions between two texts are first built based on some basic representations (e.g., word embeddings). The local interactions represented by a matching matrix is then viewed as an image, and a convolutional neural network (CNN) is employed to learn hierarchical matching patterns. Finally, the high-level matching patterns are fed into a multi-layer perceptron to produce the matching score between the two texts. The model is shown to be able to capture different levels of text matching patterns, such as n-grams and un-ordered n-terms, to improve the matching performance.

When we apply the MatchPyramid model on the ad-hoc retrieval task, we conducted extensive experiments to study the impact of different kernel sizes, pooling sizes and interaction functions on the retrieval performance. We find that three key settings are helpful for the ad-hoc retrieval task, i.e. pooling by paragraph length in document, a good similarity function which can differentiate exact matching signals from semantic matching signals, and a relative small kernel size. Finally, we show that the MatchPyramid models can significantly outperform several recently introduced deep matching models on the retrieval task, but still cannot compete with the traditional retrieval models, e.g. BM25 and language models. The recent proposed deep matching models cannot well fit the ad-hoc retrieval task right now, and more investigation need to be conducted in this direction.

2. MATCHPYRAMID

The MatchPyramid model (Figure 1) has three parts, Matching Matrix, Hierarchical Convolution and Matching Score Aggregation.

2.1 Matching Matrix

In order to keep both the word-level similarity and the matching structures, MatchPyramid introduces the Matching Matrix structure. Matching Matrix is a two-dimension structure where each element $\mathbf{M}_{ij}$ denotes the similarity between the i-th word w_i in the first piece of text and the j-th word v_j in the second piece of text.

$$\mathbf{M}_{ij} = w_i \otimes v_j, \tag{1}$$

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

Neu-IR '16 SIGIR Workshop on Neural Information Retrieval July 21, 2016, Pisa, Italy

© 2016 Copyright held by the owner/author(s).

ACM ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

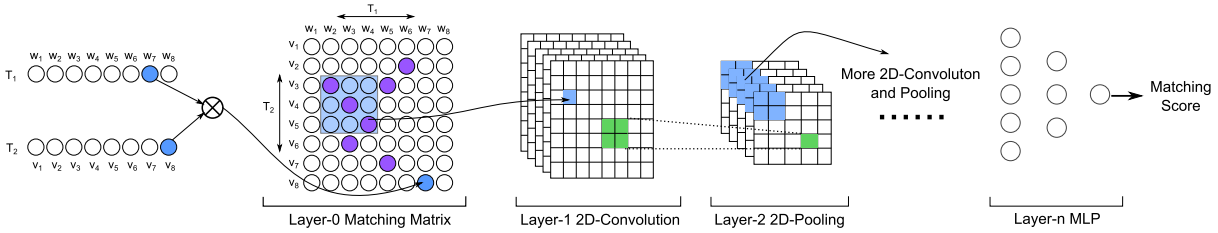


Figure 1: Model structure of MatchPyramid.

where $\otimes$ stands for a general operation to obtain the similarity.

We define four kinds of similarity functions based on words w_i and v_j, or their embeddings $\vec{\alpha_i}$ and $\vec{\beta_j}$.

Indicator Function produces either 1 or 0 to indicate whether two words are identical.

$$\mathbf{M}_{ij} = \mathbb{I}_{\{w_i = v_j\}} = \begin{cases} 1, & \text{if } w_i = v_j \\ 0, & \text{otherwise.} \end{cases} \quad (2)$$

Cosine views angles between word vectors as the similarity, and it acts as a soft indicator function.

$$\mathbf{M}_{ij} = \frac{\vec{\alpha_i}^\top \vec{\beta_j}}{\|\vec{\alpha_i}\| \cdot \|\vec{\beta_j}\|}, \quad (3)$$

where $\|\cdot\|$ stands for the ℓ_2 norm of a vector.

Dot Product further considers the norm of word vectors, as compared to cosine.

$$\mathbf{M}_{ij} = \vec{\alpha_i}^\top \vec{\beta_j}. \quad (4)$$

Gaussian Kernel is a well-known similarity function. This similarity function is introduced in this work based on our studies.

$$\mathbf{M}_{ij} = e^{-\|\vec{\alpha_i} - \vec{\beta_j}\|^2}. \quad (5)$$

We name MatchPyramid with indicator function as MP-Ind for short. Similarly, we use MP-Cos for cosine, MP-Dot for dot product and MP-Gau for Gaussian kernel.

2.2 Hierarchical Convolution

Based on the Matching Matrix, MatchPyramid conducts hierarchical convolution to extract matching patterns. Hierarchical convolution consists of convolutional layers and dynamic pooling layers, which are commonly used in CNN (such as AlexNet, GoogLeNet) for image recognition tasks.

Kernel sizes in each convolutional layer are the major hyper parameters. In text processing, the size of the kernel determines the number of words we want to compose together. In other words, the kernel size decides the maximum size of n-term features we take into account.

Besides, pooling sizes in each pooling layer are also important hyper parameters, which decide how large the area we want to take as a unit.

2.3 Matching Score Aggregation

After hierarchical convolution, two additional fully connected layers are used to aggregate the information into a single matching score. In this paper, we use 128 hidden nodes for the fully connected hidden layer and ReLU as the activation function.

2.4 Training

We use pairwise ranking loss in the training phase. Given a triple (q, d^+, d^-) where the matching score of (q, d^+) should be higher than that of (q, d^-), the loss function is define as:

$$L(q, d^+, d^-; \Theta) = \max(0, 1 - S(q, d^+) + S(q, d^-)). \quad (6)$$

where $S(q, d)$ denotes the predicted matching score for (q, d), and Θ includes the parameters for the feed forward matching network and those for the term gating network. The optimization is relatively straightforward with standard back-propagation. For regularization, we find that the early stopping [1] strategy works well for our model.

3. EXPERIMENTS

3.1 Dataset and Settings

To conduct experiments, we use TREC collection Robust04, which is a news dataset. The topics are collected from TREC Robust Track 2004. The statistics of the data collection are shown in Table 1. In this paper, the titles of the TREC topic are treat as the queries. We use the Galago Search Engine in this experiment, and both queries and documents are white-space tokenized, lower-cased, and stemmed during indexing and retrieval.

All the models share the word embeddings trained on the Wikipedia corpus under 50 dimensions. We adopt Adam algorithm [4] for model training. The learning rate is set to 10^{-4}. All the MatchPyramid models have one convolutional layer and one dynamic pooling layer, since more convolutional layers and pooling layers will lead to overfitting due to the limited training data.

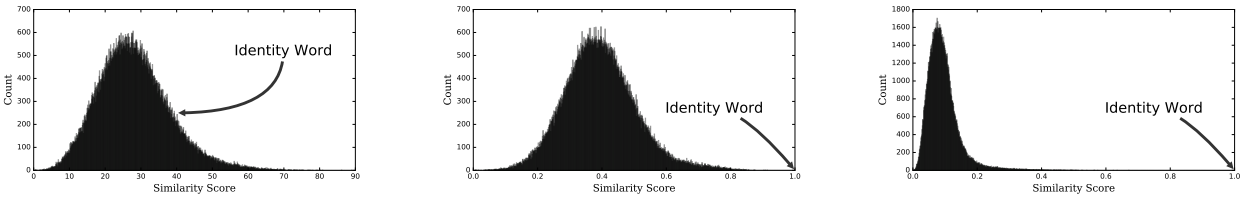
Table 1: Statistics of the TREC collection Robust04.

#Vocab	#Doc	#Query	#Retrieval doc	Avg doc
0.6M	0.5M	250	2000	477

3.2 Detailed Studies on MatchPyramid Models for Ad-hoc Retrieval

3.2.1 *Impact of Pooling Size*

We first study the effect of pooling size. Pooling layers are used to reduce the dimension of the feature maps, and to pick out the most important information for the latter layers. Especially in ac-hoc retrieval task, documents often contain hundreds of words, but most of them might be background words. So the pooling layers might be even more important to distill the useful information from the noisy background. In this experiment, we try different pooling sizes and show the results in Table 2. In our experiments, the maximum



(a) Dot Product (b) Cosine (c) Gaussian Kernel

Figure 2: Choose one word from the vocabulary and measure the similarity between other words, we draw the histogram of three type of similarity functions: dot product, cosine and gaussian kernel. The arrow point the similarity between two identity word (the word we choose).

query length is 5 and we truncate document length to 500, for the computational efficiency. Actually, we have tried other length of document, such as 1000 or full length, but the performance change is slight.

Table 2: Comparison of different pooling size of MatchPyramid. (fix kernel size 1×1)

Model	Pooling Size	MAP	nDCG@20	P@20
MP-Ind	5×100	0.175	0.320	0.254
MP-Ind	5×50	0.195	0.343	0.266
MP-Ind	5×20	0.209	0.363	0.279
MP-Ind	5×10	**0.219**	**0.387**	**0.301**
MP-Ind	5×5	0.214	0.380	0.295
MP-Ind	5×3	0.209	0.370	0.300
MP-Ind	3×20	0.213	0.357	0.285
MP-Ind	3×10	**0.225**	**0.387**	**0.302**
MP-Ind	3×5	0.225	0.385	0.301
MP-Ind	3×3	0.215	0.377	0.301
MP-Ind	1×10	0.056	0.082	0.073
MP-Ind	1×5	0.051	0.072	0.078
MP-Ind	1×3	0.043	0.066	0.053

Results show that too small or too large pooling size is not proper for this task. The best pooling size is about 3×10, which means on the query side, we use the median query length, and on the document side, we aggregate up every 50 words, close to the average length of a paragraph.

3.2.2 Impact of Similarity Function

Similarity function is used to measure the similarity of two words and build the matching matrix. For paraphrase identification task, the previous results show that indicator function is the best. For question answering, we find that dot product function is better than others. Here we try to explore the similarity function for ad-hoc retrieval task. We compare four kinds of similarity function: indicator function, dot product, cosine and gaussian kernel.

As we can see, MP-Ind performs quite well, indicating that exact matching signals are very useful for the ad-hoc retrieval task. By using embeddings, we can see gaussian kernel similarity function is better than others. To understand the underlying reasons, we first take a look at words similarity distribution as shown in Figure 2. We can see that the exact matching score (arrow pointed) is the largest value under similarity function cosine and gaussian kernel, but not under dot product. So MP-Cos and MP-Gau keep the strength of exact matching signals, while MP-Dot loses this ability and leads to worse result. The performance gap between MP-Cos and MP-Gau may be related to the gap between exact matching and the semantic matching scores. The large gap in MP-Gau indicates that the model can better differentiate exact matching from semantic matching and this leads to better performance.

Table 3: Comparison of different similarity function of MatchPyramid. (fix kernel size 1×1, pooling size 3×10)

Model	MAP	nDCG@20	P@20
MP-Ind	0.225	0.387	0.302
MP-Dot	0.095	0.149	0.142
MP-Cos	0.189	0.340	0.272
MP-Gau	**0.226**	**0.403**	**0.318**

3.2.3 Impact of Kernel Size

In this section we study the effect of kernel size of the convolutional layer in the MatchPyramid model. We conduct the experiments on different sizes of kernel with MP-Ind and MP-Gau in this section, because MP-Dot and MP-Cos do not work well in ad-hoc retrieval task. The results are shown in Table 4.

Table 4: Comparison of different kernel size of MatchPyramid. (fix pooling size 3×10)

Model	Kernel Size	MAP	nDCG@20	P@20
MP-Ind	1×1	0.225	**0.387**	**0.302**
MP-Ind	1×3	**0.226**	0.382	0.294
MP-Ind	1×5	0.223	0.382	0.297
MP-Ind	3×3	0.221	0.379	0.295
MP-Ind	5×5	0.219	0.378	0.295
MP-Gau	1×1	0.226	0.403	0.318
MP-Gau	1×3	**0.232**	**0.411**	**0.327**
MP-Gau	1×5	0.226	0.409	0.326
MP-Gau	3×3	0.220	0.400	0.312
MP-Gau	5×5	0.201	0.371	0.301

We have tried two kinds of kernel sizes, $1 \times n$ and $n \times n$, where $n \in [1, 3, 5]$. We expect kernel $1 \times n$ to capture the information around the central word in the kernel window. The results show that different kernel sizes under indicator similarity function perform similarly. It is reasonable if we look at the Matching Matrix generated by the indicator function. The Matching Matrix is very sparse and in

a kernel window there is usually one non-zero element. So the kernel size is not that important for indicator function. However, by using the gaussian kernel, we introduce semantic word similarity into the model, and a proper kernel size will get more information and generate a better result. We find that MP-Gau with kernel size 1×3 achieves the best performance. Additionally, we expect kernel $n \times n$ to capture the word proximity information, such as n-gram matching. Surprisingly there is no improvement in these experiments. The reason might be that the dataset is too small to learn the complex proximity patterns.

3.3 Comparison with Baseline Models

We further compare the MatchPyramid model with a set of baseline models. We adopt three types of baselines, including traditional models, representation-based deep matching models and interaction-based deep matching model.

Traditional models include:

QL: Query likelihood model based on Dirichlet smoothing[11] is one of the best language models.

BM25: Based on BM25 formula [8], is another highly effective retrieval model.

Representation-based deep matching models include:

DSSM: DSSM model [3] uses fully connected layers to encode query and document into two fix-length vectors, then uses cosine similarity to compute the matching score. Since DSSM needs large scale training data due to its huge parameter size, we directly used the released model[1] (trained on large click-through dataset) in our experiments.

CDSSM: CDSSM model [9] is similar with DSSM, but use convolutional layers to encode query and document. For the same reason as DSSM, we also made use of the released model directly.

ARC-I: ARC-I model [2] uses convolutional layers to encode two texts, and uses fully connected layers to aggregate matching score.

Interaction-based deep matching models include:

ARC-II: ARC-II [2] constructs local interactions by adding up word embeddings in a small context window, then makes use of convolutional layers to extract features from the interactions.

Table 5: Comparison of different retrieval models over the TREC collection Robust04. † models trained on large click-through dataset.

Type	Name	MAP	nDCG@20	P@20
Traditional	QL	0.253	0.415	0.369
Model	BM25	**0.255**	**0.418**	**0.370**
Representation	DSSM†	0.095	0.201	0.171
Based Model	CDSSM†	0.067	0.146	0.125
	ARC-I	0.041	0.066	0.065
Interaction	ARC-II	0.067	0.147	0.128
Based Model	MP-Gau	0.232	0.411	0.327

The experimental results (see Table 5) show that MatchPyramid outperforms all the representation-based deep matching models. The major reason is that it can retain all the low-level matching signals which are important for ad-hoc retrieval. For ARC-II, using 1D convolution to generate intercations signals in a small context window seems not very effective for matching. However, we find that the best performing deep matching model, MP-Gau, still cannot compete with traditional retrieval models. The results indicate that the ad-hoc retrieval task may be quire different from other text matching tasks, such as paraphrase identification and question answering. We need some further studies on the differences between these tasks for designing better deep matching models.

[1]http://research.microsoft.com/en-us/downloads/731572aa-98e4-4c50-b99d-ae3f0c9562b9/

4. CONCLUSIONS

In this paper, we apply MatchPyramid model to ad-hoc retrieval task and discuss the impact of different kernel sizes, pooling sizes and similarity functions. We find that pooling by paragraph length in document, a good similarity function which can differentiate exact matching signals from semantic matching signals, and a relative small kernel size are helpful for the ad-hoc retrieval task. Experiments show that the MatchPyramid models can significantly outperform several recently introduced deep matching models on the retrieval task, but still cannot compete with the traditional retrieval models. These results encourage us to seek deeper understanding of the text matching task in ad-hoc retrieval and propose better models accordingly.

5. REFERENCES

[1] R. C. S. L. L. Giles. Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping. In *Advances in Neural Information Processing Systems 13: Proceedings of the 2000 Conference*, volume 13, page 402. MIT Press, 2001.

[2] B. Hu, Z. Lu, H. Li, and Q. Chen. Convolutional neural network architectures for matching natural language sentences. In *Advances in Neural Information Processing Systems*, pages 2042–2050, 2014.

[3] P.-S. Huang, X. He, J. Gao, L. Deng, A. Acero, and L. Heck. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*, pages 2333–2338. ACM, 2013.

[4] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[5] Z. Lu and H. Li. A deep architecture for matching short texts. In *Advances in Neural Information Processing Systems*, pages 1367–1375, 2013.

[6] L. Pang, Y. Lan, J. Guo, J. Xu, S. Wan, and X. Cheng. Text matching as image recognition. *CoRR*, abs/1602.06359, 2016.

[7] X. Qiu and X. Huang. Convolutional neural tensor network architecture for community-based question answering. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1305–1311, 2015.

[8] S. E. Robertson and S. Walker. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 232–241. Springer-Verlag New York, Inc., 1994.

[9] Y. Shen, X. He, J. Gao, L. Deng, and G. Mesnil. 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, pages 101–110. ACM, 2014.

[10] R. Socher, E. H. Huang, J. Pennin, C. D. Manning, and A. Y. Ng. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Advances in Neural Information Processing Systems*, pages 801–809, 2011.

[11] C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to ad hoc information retrieval. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 334–342. ACM, 2001.