Text Matching as Image Recognition

Liang Pang[*], **Yanyan Lan**[†], **Jiafeng Guo**[†], **Jun Xu**[†], **Shengxian Wan**[*], and **Xueqi Cheng**[†]

CAS Key Laboratory of Network Data Science and Technology,
Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China
[*]{pangliang,wanshengxian}@software.ict.ac.cn, [†]{lanyanyan,guojiafeng,junxu,cxq}@ict.ac.cn

Abstract

Matching two texts is a fundamental problem in many natural language processing tasks. An effective way is to extract meaningful matching patterns from words, phrases, and sentences to produce the matching score. Inspired by the success of convolutional neural network in image recognition, where neurons can capture many complicated patterns based on the extracted elementary visual patterns such as oriented edges and corners, we propose to model text matching as the problem of image recognition. Firstly, a matching matrix whose entries represent the similarities between words is constructed and viewed as an image. Then a convolutional neural network is utilized to capture rich matching patterns in a layer-by-layer way. We show that by resembling the compositional hierarchies of patterns in image recognition, our model can successfully identify salient signals such as n-gram and n-term matchings. Experimental results demonstrate its superiority against the baselines.

Introduction

Matching two texts is central to many natural language applications, such as machine translation (Brown et al. 1993), question and answering (Xue, Jeon, and Croft 2008), paraphrase identification (Socher et al. 2011) and document retrieval (Li and Xu 2014). Given two texts $T_1 = (w_1, w_2, \ldots, w_m)$ and $T_2 = (v_1, v_2, \ldots, v_n)$, the degree of matching is typically measured as a score produced by a scoring function on the representation of each text:

$$\text{match}(T_1, T_2) = \text{F}(\Phi(T_1), \Phi(T_2)), \quad (1)$$

where w_i and v_j denotes the i-th and j-th word in T_1 and T_2, respectively. Φ is a function to map each text to a vector, and F is the scoring function for modeling the interactions between them.

A successful matching algorithm needs to capture the rich interaction structures in the matching process. Taking the task of paraphrase identification for example, given the following two texts:

T_1 : *Down the ages noodles and dumplings were famous Chinese food.*

T_2 : *Down the ages dumplings and noodles were popular in China.*

We can see that the interaction structures are of different levels, from words, phrases to sentences. Firstly, there are many word level matching signals, including identical word matching between "*down*" in T_1 and "*down*" in T_2, and similar word matching between "*famous*" in T_1 and "*popular*" in T_2. These signals compose phrase level matching signals, including n-gram matching between "*down the ages*" in T_1 and "*down the ages*" in T_2, unordered n-term matching between "*noodles and dumplings*" in T_1 and "*dumplings and noodles*" in T_2, and semantic n-term matching between "*were famous Chinese food*" in T_1 and "*were popular in China*" in T_2. They further form sentence level matching signals, which are critical for determining the matching degree of T_1 and T_2. How to automatically find and utilize these hierarchical interaction patterns remains a challenging problem.

In image recognition, it has been widely observed that the convolutional neural network (CNN) (LeCun et al. 1998; Simard, Steinkraus, and Platt 2003) can successfully abstract visual patterns from raw pixels with layer-by-layer composition (Girshick et al. 2014). Inspired by this observation, we propose to view text matching as image recognition and use CNN to solve the above problem. Specifically, we first construct a word level similarity matrix, namely matching matrix, to capture the basic word level matching signals. The matching matrix can be viewed as: 1) a binary image if we define the similarity to be 0-1, indicating whether the two corresponding words are identical; 2) a gray image if we define the similarity to be real valued, which can be achieved by calculating the cosine or inner product based on the word embeddings. Then we apply a convolutional neural network on this matrix. Meaningful matching patterns such as n-gram and n-term can therefore be fully captured within this architecture. We can see that our model takes text matching as a multi-level abstraction of interaction patterns between words, phrases and sentences, with a layer-by-layer architecture, so we name it MatchPyramid.

The experiments on the task of paraphrase identification show that MatchPyramid (with 0-1 matching matrix) outperforms the baselines, by solely leveraging interactions between texts. While for other tasks such as paper citation matching, where semantic is somehow important, Match-

Pyramid (with real-valued matching matrix) performs the best by considering both interactions and semantic representations.

Contributions of this paper include: 1) a novel view of text matching as image recognition; 2) the proposal of a new deep architecture based on the matching matrix, which can capture the rich matching patterns at different levels, from words, phrases, to the whole sentences; 3) experimental analysis on different tasks to demonstrate the superior power of the proposed architecture against competitor matching algorithms.

Motivation

It has been widely recognized that making a good matching decision requires to take into account the rich interaction structures in the text matching process, starting from the interactions between words, to various matching patterns in the phrases and the whole sentences. Taking the aforementioned two sentences as an example, the interaction structures are of different levels, as illustrated in Figure 1.
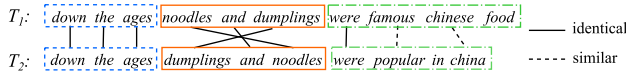

Figure 1: An example of interaction structures in paraphrase identification.

Word Level Matching Signals refer to matchings between words in the two texts, including not only identical word matchings, such as "*down–down*", "*the–the*", "*ages–ages*", "*noodles–noodles*", "*and–and*","*dumplings–dumplings*" and "*were–were*", but also similar word matchings, such as "*famous–popular*" and "*chinese–china*".

Phrase Level Matching Signals refer to matchings between phrases, including n-gram and n-term. N-gram matching occurs with n exactly matched successive words, e.g. "(*down the ages*)–(*down the ages*)". While n-term matching allows for order or semantic alternatives, e.g. "(*noodles and dumplings*)–(*dumplings and noodles*)", and "(*were famous chinese food*)–(*were popular in china*)".

Sentence Level Matching Signals refer to matchings between sentences, which are composed of multiple lower level matching signals, e.g. the three successive phrase level matchings mentioned above. When we consider matchings between paragraphs that contain multiple sentences, the whole paragraph will be viewed as a long sentence and the same composition strategy would generate paragraph level matching signals.

To sum up, the interaction structures are compositional hierarchies, in which higher level signals are obtained by composing lower level ones. This is similar to image recognition. In an image, raw pixels provide basic units of the image, and each patch may contain some elementary visual features such as oriented edges and corners. Local combinations of edges form motifs, motifs assemble into parts, and parts form objects. We give an example to show the relationships between text matching and image recognition (Jia et al. 2014), as illustrated in Figure 2. In the area of image recognition, CNN has been recognized as one the most successful

way to capture different levels of patterns in image (Zeiler and Fergus 2014). Therefore, it inspires us to transform text matching to image recognition and employ CNN to solve it. However, the representations of text and image are so different that it remains a challenging problem to perform such transformation.

MatchPyramid

In this section we introduce a new deep architecture for text matching, namely MatchPyramid. The main idea comes from modeling text matching as image recognition, by taking the matching matrix as an image, as illustrated in Figure 3.

Matching Matrix: Bridging the Gap between Text Matching and Image Recognition

As discussed before, one challenging problem by modeling text matching as image recognition lies in the different representations of text and image: the former are two 1D (one-dimensional) word sequences while the latter is typically a 2D pixel grid. To address this issue, we represent the input of text matching as a matching matrix $\mathbf{M}$, with each element $\mathbf{M}_{ij}$ standing for the basic interaction, i.e. similarity between word w_i and v_j (see Eq. 2). Here for convenience, w_i and v_j denotes the i-th and j-th word in two texts respectively, and $\otimes$ stands for a general operator to obtain the similarity.

$$\mathbf{M}_{ij} = w_i \otimes v_j. \tag{2}$$

In this way, we can view the matching matrix $\mathbf{M}$ as an image, where each entry (i.e. the similarity between two words) stands for the corresponding pixel value. We can adopt different kinds of $\otimes$ to model the interactions between two words, leading to different kinds of raw images. In this paper, we give three examples as follows.

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

$$\mathbf{M}_{ij} = \mathbb{I}_{\{w_i = v_j\}} = \begin{cases} 1, & \text{if } w_i = v_j \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

One limitation of the indicator function is that it cannot capture the semantic matching between similar words. To tackle this problem, we define $\otimes$ based on word embeddings, which will make the matrix more flexible to capture semantic interactions. Given the embedding of each word $\vec{\alpha}_i = \Phi(w_i)$ and $\vec{\beta}_j = \Phi(v_j)$, which can be obtained by recent Word2Vec (Mikolov et al. 2013) technique, we introduce the other two operators: cosine and dot product.

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

$$\mathbf{M}_{ij} = \frac{\vec{\alpha}_i^\top \vec{\beta}_j}{\|\vec{\alpha}_i\| \cdot \|\vec{\beta}_j\|}, \tag{4}$$

where $\| \cdot \|$ stands for the norm of a vector, and ℓ_2 norm is used in this paper.

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

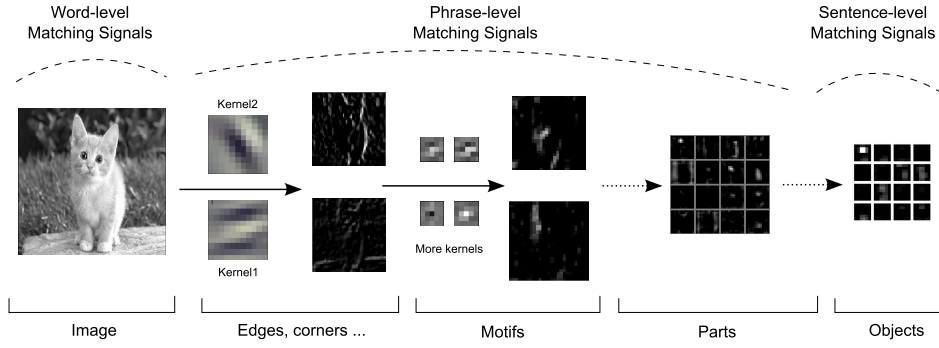$$\mathbf{M}_{ij} = \vec{\alpha}_i^\top \vec{\beta}_j. \tag{5}$$

Figure 2: Relationships between text matching and image recognition.
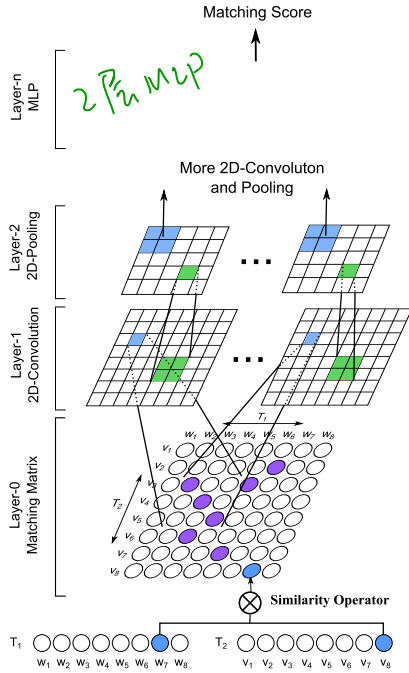

Figure 3: An overview of MatchPyramid on Text Matching.

Based on these three different operators, the matching matrices of the given example are shown in Fig 4. Obviously we can see that Fig 4(a) corresponds to a binary image, and Fig 4(b) correspond to gray images.

Hierarchical Convolution: A Way to Capture Rich Matching Patterns

The body of MatchPyramid is a typical convolutional neural network, which can extract different levels of matching patterns. For the first layer of CNN, the k-th kernel $\mathbf{w}^{(1,k)}$ scans over the whole matching matrix $\mathbf{z}^{(0)} = \mathbf{M}$ to generate a feature map $\mathbf{z}^{(1,k)}$:

$$\mathbf{z}_{i,j}^{(1,k)} = \sigma\left(\sum_{s=0}^{r_k-1}\sum_{t=0}^{r_k-1} \mathbf{w}_{s,t}^{(1,k)} \cdot \mathbf{z}_{i+s,j+t}^{(0)} + b^{(1,k)}\right), \quad (6)$$

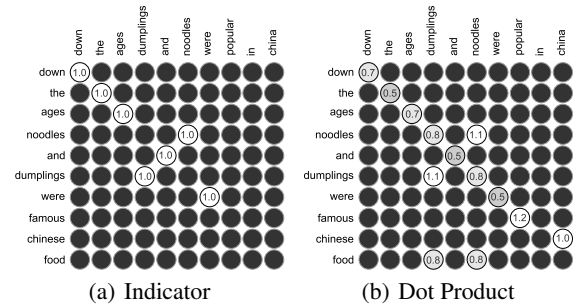
(a) Indicator   (b) Dot Product

Figure 4: Three different matching matrices, where solid circles elements are all valued 0.

where r_k denotes the size of the k-th kernel. In this paper we use square kernel, and ReLU (Dahl, Sainath, and Hinton 2013) is adopted as the active function σ.

Dynamic pooling strategy (Socher et al. 2011) is then used to deal with the text length variability. By applying dynamic pooling, we will get fixed-size feature maps:

$$\mathbf{z}_{i,j}^{(2,k)} = \max_{0 \le s < d_k} \max_{0 \le t < d_k'} \mathbf{z}_{i \cdot d_k + s, j \cdot d_k' + t}^{(1,k)}, \quad (7)$$

where d_k and d_k' denote the width and length of the corresponding pooling kernel, which are determined by the text lengths n and m, and output feature map size $n' \times m'$, i.e. $d_k = \lceil n/n' \rceil, d_k' = \lceil m/m' \rceil$.

After the first convolution and dynamic pooling, we continue to obtain higher level features $\mathbf{z}^{(l)}, l \ge 2$ by further convolution and max-pooling, with general formulations:

$$\mathbf{z}_{i,j}^{(l+1,k')} = \sigma\left(\sum_{k=0}^{c_l-1}\sum_{s=0}^{r_k-1}\sum_{t=0}^{r_k-1} \mathbf{w}_{s,t}^{(l+1,k')} \cdot \mathbf{z}_{i+s,j+t}^{(l,k)} + b^{(l+1,k)}\right),$$
$$l = 2, 4, 6, \ldots, \quad (8)$$

$$\mathbf{z}_{i,j}^{(l+1,k)} = \max_{0 \le s < d_k} \max_{0 \le t < d_k} \mathbf{z}_{i \cdot d_k + s, j \cdot d_k + t}^{(l,k)},$$
$$l = 3, 5, 7, \ldots, \quad (9)$$

where c_l denote the number of feature maps in the l-th layer.

Analysis of Hierarchical Convolution

Similar to CNN in image recognition where it can make abstractions based on extracted elementary visual patterns
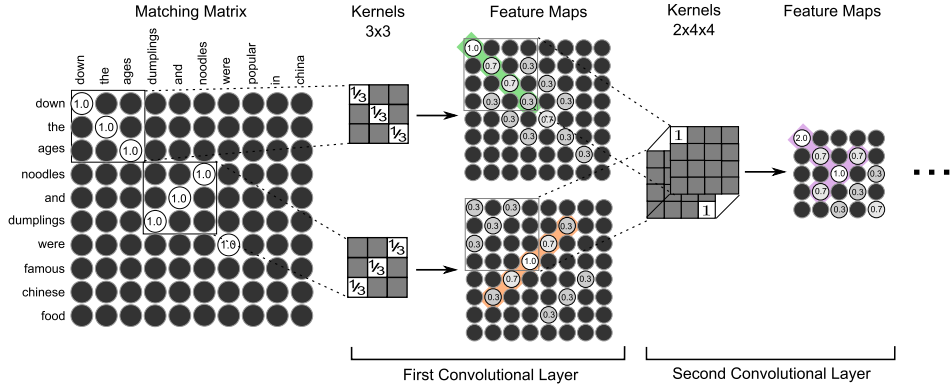
Figure 5: An illustration of Hierarchical Convolution.

such as oriented edges and corners, the hierarchical convolution in MatchPyramid can also capture important phrase level interactions from word level matching and make further compositions. We revisit our example, and show how it works[1], as illustrated in Figure 5.

(1) With the given two kernels, we can see clearly that the first convolutional layer can capture both n-gram matching signals "(down the ages)–(down the ages)" and n-term matching signal "(noodles and dumplings)–(dumplings and noodles)". The extracted matching patterns are like edges in image recognition (refer Figure 2).

(2) The following convolutional layers make compositions and form higher level of matching patterns. For example, from the second layer, we can see that a more complicated "T-type" pattern captured with the given 3D kernel. It looks like some motif (parts) obtained in image recognition (refer Figure 2).

From the above analysis we can see that MatchPyramid can abstract complicated matching patterns, from phrase to sentence level, by hierarchical convolution.

Matching Score and Training

We use a MLP (Multi-Layer Perception) to produce the final matching score. Take binary classification and two-layer perceptron for example, we will obtain a 2-dimensional matching score vector:

$$(s_0, s_1)^\top = \mathbf{W}_2 \sigma \big(\mathbf{W}_1 \mathbf{z} + \mathbf{b}_1 \big) + \mathbf{b}_2, \qquad (10)$$

where s_0 and s_1 are the matching scores of the corresponding class, $\mathbf{z}$ is the output of the hierarchical convolution, $\mathbf{W}_i$ is the weight of the i-th MLP layer and σ denotes the activation function.

Softmax function is utilized to output the probability of belonging to each class, and cross entropy is used as the objective function for training. Therefore the optimization

---

[1]Here we take the matching matrix with indicator function as example, similar observations can be obtained for other matching matrix with cosine similarity and dot product.

becomes minimizing:

$$loss = -\sum_{i=1}^{N} \Big[y^{(i)} \log(p_1^{(i)}) + (1 - y^{(i)}) \log(p_0^{(i)}) \Big],$$
$$\qquad\qquad\qquad\qquad\qquad\qquad (11)$$
$$p_k = \frac{e^{s_k}}{e^{s_0} + e^{s_1}}, \quad k = 0, 1,$$

where $y^{(i)}$ is the label of the i-th training instance. The optimization is relatively straightforward with the standard back-propagation (Williams and Hinton 1986). We apply stochastic gradient descent method Adagrad (Duchi, Hazan, and Singer 2011) for the optimization of models. It performs better when we use the mini-batch strategy (32∼50 in size), which can be easily parallelized on single machine with multi-cores. For regularization, we find that some common strategies like early stopping (Giles 2001) and dropout (Hinton et al. 2012) are enough for our model.

Experiments

In this section, we conduct experiments on two tasks, i.e. paraphrase identification and paper citation matching to demonstrate the superiority of MatchPyramid against baselines.

Competitor Methods and Experimental Settings

ALLPOSITIVE: All of the test data are predicted as positive.

TF-IDF: TF-IDF (Salton, Fox, and Wu 1983) is a widely used method in text mining. In this method, each text is represented as a $|V|$-dimensional vector with each element stands for the TF-IDF score of the corresponding word in the text, where $|V|$ is the vocabulary size. In this paper, idf score is calculated in the whole dataset. The final matching score is produced by the inner product of the two vectors.

DSSM/CDSSM: Since DSSM (Huang et al. 2013) and CDSSM (Gao et al. 2014; Shen et al. 2014) need large data for training, we directly use the released models[2] (trained on large click-through dataset) on our test data.

---

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

ARC-I/ARC-II: We implement ARC-I and ARC-II (Hu et al. 2014) due to there is no publicly available codes, using exactly the same setting as described in the original paper.

There are three versions of MatchPyramid, depending on different methods used for constructing the matching matrices, denoted as MP-IND, MP-COS, and MP-DOT, respectively. All these models use two convolutional layers, two max-pooling layers (one of which is a dynamic pooling layer for variable length) and two full connection layers. The number of feature maps is 8 and 16 for the first and second convolutional layer, respectively. While the kernel size is set to be 5×5 and 3×3, respectively. Unlike ARC-II which initiates with `Word2Vec` trained on Wikipedia, we initiate the word vectors in MP-COS and MP-DOT randomly from a unit ball. Thus our model do not require any external sources.

Experiment I: Paraphrase Identification

Paraphrase identification aims to determine whether two sentences have the same meaning, a problem considered as a touchstone of natural language understanding. Here we use the benchmark MSRP dataset (Dolan and Brockett 2005), which contains 4076 instances for training and 1725 for testing. The experimental results are listed in Table 1. We can

Table 1: Results on MSRP.

Model	Acc.(%)	F_1(%)
ALLPOSITIVE	66.50	79.87
TF-IDF	70.31	77.62
DSSM	70.09	80.96
CDSSM	69.80	80.42
ARC-I	69.60	80.27
ARC-II	69.90	80.91
MP-IND	75.77	82.66
MP-COS	75.13	82.45
MP-DOT	**75.94**	**83.01**

see that traditional simple model such as TF-IDF has already achieved a high accuracy of about 70%, though it only uses the unigram matching signals. Our methods performs much better than TF-IDF, which indicates that the complicated matching patterns captured by hierarchical convolution are important to the text matching task. For the comparison with recent deep models, we can see that DSSM performs better than the others (though the improvement is quite limited), and our models (MP-IND, MP-COS and MP-DOT) outperform all of them. Though the best performance of our model (75.94%/83.01%) is still slightly worse than URAE (Socher et al. 2011) (76.8%/83.6%), URAE relies heavily on pretraining with an external large dataset annotated with parse tree information. In the future work, we will study how to utilize external data to further improve our models.

We also visualize what we have learned in MatchPyramid[3], with expectation that we can gain some insights from

---

[3]Here we only demonstrate the case of MP-DOT due to space limitation. Similar results can be observed with MP-IND and MP-COS.

the process. Specifically, we take a pair of texts as an example (selected from the MSRP dataset), and illustrate the feature maps and kernels in Figure 6. We can see that n-gram and n-term matching, which are emphasized in the blue and yellow color in original texts, are represented as a diagonal sub-matrix emphasized with the blue and yellow rectangles in the matching matrix, respectively. Kernel 1 and kernel 2 are the two kernels learned in the first convolutional layer, which well captures the important n-gram and n-term matching signals respectively. We can see that these patterns are quite similar to the edge extracted by CNN in image recognition (see Figure 2). We also give some more kernels and show some patterns learned in the second convolutional layer. We can see that the latter layer make compositions and keep the useful matching signals until passing it to the MLP classifier. This explains clearly why our model works well: MatchPyramid captures useful matching patterns at different levels, from words, phrase, to sentences, with a similar process in image recognition.

Experiment II: Paper Citation Matching

We evaluate the effectiveness of MatchPyramid with another text matching task called paper citation matching, based on a large academic dataset[4]. Basically, we are given a set of papers along with their abstracts. A paper and its citations' abstracts then becomes a pair of texts, and defined as a type of matching. One representative example is given as follows:

T_1 : *this article describes pulsed thermal time of flight ttof flow sensor system as two subsystems pulsed wire system and heat flow system the entire flow sensor is regarded system theoretically as linear.*

T_2 : *the authors report on novel linear time invariant lti modeling of flow sensor system based on thermal time of flight tof principle by using pulsed hot wire anemometry thermal he at pulses.*

We can see that the matching here should take both lexical and semantic information into consideration. The dataset is collected from a commercial academic website. It contains 838 908 instances (text pairs) in total, where there are 279 636 positive (matched) instances and 559 272 negative (mismatch) instances. The negative instances are randomly sampled papers which have no citation relations. We split the whole dataset into three parts, 599 196 instances for training, 119 829 for validation and 119 883 for testing.

The results in Table 2 show that TF-IDF is also a strong baseline on this dataset, which is even better than some deep models such as DSSM and CDSSM. This may be caused by the large difference between the testing data (paper citation data) and training data (click-through data) used in DSSM and CDSSM. ARC-I and ARC-II gain a significant improvement over these models, which may benefit much from the large training data. As for our models, the best performance is still achieved by MP-DOT (88.73%/82.86%), which is better than ARC-II (86.84%/79.57%). MP-COS also gains a better result than ARC-II. The reason of the poor performance of MP-IND on this task may lie in that the indicator

---

[4]We only use the first 32 words in the abstract.

T_1: PCCW's chief operating officer, Mike Butcher, and Alex Arena, the chief financial officer, will report directly to Mr So.

T_2: Current Chief Operating Officer Mike Butcher and Group Chief Financial Officer Alex Arena will report to So.
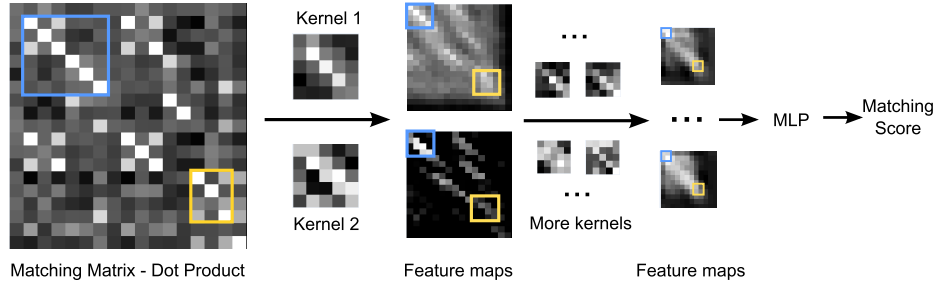

Figure 6: Analysis of the feature maps and kernels in the MatchPyramid Model. The brighter the pixel is, the larger value it has. Better viewed in color.

Table 2: Results on the task of paper citation matching.

Model	Acc.(%)	F_1(%)
ALLPOSITIVE	33.33	50.00
TF-IDF	82.63	70.21
DSSM	71.97	29.88
CDSSM	69.84	19.97
ARC-I	84.51	76.79
ARC-II	86.48	79.57
MP-IND	73.76	44.71
MP-COS	86.65	79.70
MP-DOT	**88.73**	**82.86**

function only captures the exact matching between words, but omits the semantic similarity.

Table 3: The norm of learned word embeddings on the task of paper citation matching.

Word	the	with	for	be	are
Len	0.448	0.508	0.509	0.510	0.515
Word	robotics	java	snakes	musical	rfid
Len	1.572	1.576	1.589	1.610	1.878

We further show the reason why MP-DOT performs better than MP-COS by analyzing the learned word embeddings. Specifically, we pick some words with large and small norm, listed in Table 3. We can see that most words with small norm are indeed useless for matching, while most words with large norm (such as *robotics* and *java*) are domain terms which play an important role in paper citation matching. By further considering the importance of words, MP-DOT can capture more semantic information than MP-COS and thus achieve better performance.

Related Work

Most previous work on text matching tries to find good representations for a single text, and usually use a simple scoring function to obtain the matching results. Examples include Partial Least Square (Wu, Li, and Xu 2013), Canonical Correlation Analysis (Hardoon and Shawe-Taylor 2003)

and some deep models such as DSSM (Huang et al. 2013), CDSSM (Gao et al. 2014; Shen et al. 2014) and ARC-I (Hu et al. 2014).

Recently, a brand new approach focusing on modeling the interaction between two sentences has been proposed and gained much attention, examples include DEEPMATCH (Lu and Li 2013), URAE (Socher et al. 2011) and ARC-II (Hu et al. 2014). Our model falls into this category, thus we give some detailed discussions on the differences of our model against these methods.

DEEPMATCH uses topic model to construct the interactions between two texts, and then make different levels of abstractions by a hierarchical architecture based on the relationships between topics. Compared with our matching matrix defined at word level, DEEPMATCH uses topic information with more rough granularity. Moreover, it relies largely on the quality of learned topic model, and the hierarchies are usually ambiguous since the relationships between topics are not absolute. On the contrary, MatchPyramid clearly models the interactions at different levels, from words, phrases to sentences.

URAE constructs the interactions between two texts based on the syntactic trees, thus it relies on a predefined compact vectorial representation of text. Specifically, URAE first learns the representation of each node on the tree by a auto-encoder, then directly inserts different levels of interaction, such as word, prase and sentence, to a single matrix. Different from that, our MatchPyramid is end-to-end, and captures different levels of interactions in a hierarchical way.

ARC-II and ARC-I are both proposed based on convolutional sentence model DCNN (Kalchbrenner, Grefenstette, and Blunsom 2014). Different from ARC-I which defers the interaction of two texts to the end of the process, ARC-II lets them meet early by directly interleaving them to a single representation, and makes abstractions on this basis. Therefore, ARC-II is capturing sentence level interactions directly. However, it is not clear what exactly the interactions are, since they used a sum operation. Our model is also based on a convolutional neural network, but the idea is quite different from that of ARC-II. It is clear that we start from word level matching patterns, and compose to phrase and sentence level matching pattern layer by layer.

Conclusion

In this paper, we view text matching as image recognition, and propose a new deep architecture, namely MatchPyramid. Our model can automatically capture important matching patterns such as unigram, n-gram and n-term at different levels. Experimental results show that our model can outperform baselines, including some recently proposed deep matching algorithms.

Acknowledgments

References

Brown, P. F.; Pietra, V. J. D.; Pietra, S. A. D.; and Mercer, R. L. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational linguistics* 19(2):263–311.

Dahl, G. E.; Sainath, T. N.; and Hinton, G. E. 2013. Improving deep neural networks for lvcsr using rectified linear units and dropout. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, 8609–8613. IEEE.

Dolan, W. B., and Brockett, C. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proc. of IWP*.

Duchi, J.; Hazan, E.; and Singer, Y. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research* 12:2121–2159.

Gao, J.; Pantel, P.; Gamon, M.; He, X.; Deng, L.; and Shen, Y. 2014. Modeling interestingness with deep neural networks. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*.

Giles, R. C. S. L. L. 2001. 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, 402. MIT Press.

Girshick, R.; Donahue, J.; Darrell, T.; and Malik, J. 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, 580–587. IEEE.

Hardoon, D. R., and Shawe-Taylor, J. 2003. Kcca for different level precision in content-based image retrieval. In *Proceedings of Third International Workshop on Content-Based Multimedia Indexing, IRISA, Rennes, France*.

Hinton, G. E.; Srivastava, N.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR* abs/1207.0580.

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

Huang, P.-S.; He, X.; Gao, J.; Deng, L.; Acero, A.; and Heck, L. 2013. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM international conference on Conference on Information and Knowledge Management*, 2333–2338. ACM.

Jia, Y.; Shelhamer, E.; Donahue, J.; Karayev, S.; Long, J.; Girshick, R.; Guadarrama, S.; and Darrell, T. 2014. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*.

Kalchbrenner, N.; Grefenstette, E.; and Blunsom, P. 2014. A convolutional neural network for modelling sentences. *CoRR* abs/1404.2188.

LeCun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86(11):2278–2324.

Li, H., and Xu, J. 2014. Semantic matching in search. *Foundations and Trends in Information Retrieval* 7(5):343–469.

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

Mikolov, T.; Chen, K.; Corrado, G.; and Dean, J. 2013. Efficient estimation of word representations in vector space. *CoRR* abs/1301.3781.

Salton, G.; Fox, E. A.; and Wu, H. 1983. Extended boolean information retrieval. *Communications of the ACM* 26(11):1022–1036.

Shen, Y.; He, X.; Gao, J.; Deng, L.; and Mesnil, G. 2014. 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*, 101–110. ACM.

Simard, P. Y.; Steinkraus, D.; and Platt, J. C. 2003. Best practices for convolutional neural networks applied to visual document analysis. In *2013 12th International Conference on Document Analysis and Recognition*, volume 2, 958–958. IEEE Computer Society.

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

Williams, D. R. G. H. R., and Hinton, G. 1986. Learning representations by back-propagating errors. *Nature* 323–533.

Wu, W.; Li, H.; and Xu, J. 2013. Learning query and document similarities from click-through bipartite graph with metadata. In *Proceedings of the sixth ACM international conference on WSDM*, 687–696. ACM.

Xue, X.; Jeon, J.; and Croft, W. B. 2008. Retrieval models for question and answer archives. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, 475–482. ACM.

Zeiler, M. D., and Fergus, R. 2014. Visualizing and understanding convolutional networks. In *Computer Vision–ECCV 2014*. Springer. 818–833.