POLAR: Attention-based CNN for One-shot Personalized Article Recommendation

Zhengxiao Du[1]✉, Jie Tang[1], and Yuhui Ding[1]

Department of Computer Science and Technology, Tsinghua University, China
duzx16@mails.tsinghua.edu.cn, jietang@tsinghua.edu.cn,
dingyh15@mails.tsinghua.edu.cn

Abstract. In this paper, we propose POLAR, an attention-based CNN combined with one-shot learning for personalized article recommendation. Given a query, POLAR uses an attention-based CNN to estimate the relevance score between the query and related articles. The attention mechanism can help significantly improve the relevance estimation. For example, on AMiner, this can help achieve a +5.0% improvement in terms of NDCG@3. One more challenge in personalized article recommendation is how to collect statistically sufficient training data for a recommendation model. POLAR combines a one-shot learning function into the recommendation model, which further gains significant improvements. For example, on AMiner, with only 1.6 feedbacks on average, POLAR achieves 2.7% improvement by NDCG@3. We evaluate the proposed POLAR on three different datasets: AMiner, Patent, and RARD. Experimental results demonstrate the effectiveness of the proposed model. Recently, we have successfully deployed POLAR into AMiner as the recommendation engine for article recommendation, which further confirms the effectiveness of the proposed model.

Keywords: Personalized recommendation · Term weighting · CNN · One-shot learning

1 Introduction

Nowadays the amount of academic articles has been quite large and increases dramatically every year. According to the statistics from NCSES[1], global publication output per year in science and engineering grew at an average annual rate of 6% from 2004 to 2014. How to recommend to users the articles they are most interested in has become a key problem for digital library service providers. Many academic search sites provide article recommendation on the information page of a specific article to help users find related articles. However, these recommendations are often based on keyword similarity between the current article and candidates, which doesn't contain any form of personalization.

Typically, an article covers several different topics. For example, this paper, as the keywords show, covers *Personalized Recommendation*, *Term Weighting*,

[1] https://www.nsf.gov/statistics/

CNN and *One-shot Learning.* Users with different backgrounds and interests may prefer articles related to different topics. Recommendation results which ignore personalization don't take user diversity into account, and cannot satisfy most users.

Good personalization can be challenging. For academic search sites, many users are cold-start users, whose profiles are incomplete or missing and cannot provide much helpful information. Methods based on user feedback are typically preferred. However, the user feedback can be quite sparse and implicit. For new users only implicit feedback from the same session is available. Therefore, it is difficult to apply the traditional recommendation methods such as Content-based Recommendation [21] or Collaborative Filtering [2].

We define the personalized article recommendation problem as follows.

Definition 1. *Let $D = \{d_1, d_2, \cdots, d_N\}$ denote the set of candidate articles, where N is the candidate size. The input of our problem is a query article d_q, and a support set $S = \{(\hat{d}_i, \hat{y}_i)\}_{i=1}^{T}$ related to user u, where $\hat{d}_i$ is a support article and $\hat{y}_i$ represents the user feedback for $\hat{d}_i$. The output is a totally ordered set $R(d_q, S) \subset D$ with $|R| = k$, which is the top-k recommendation for u with respect to d_q.*

Text similarity, which plays a key role in recommender systems and information retrieval, poses another challenge. The bag-of-words model, on which most traditional methods are based, discards the information about word order and cooccurrence. Therefore these methods cannot capture the matching signals in phrase or higher levels. Recently, due to the development of word embeddings and neural networks, many neural similarity models that can directly deal with word sequences are proposed [20] [29], but they often treat all the words in an article indiscriminately. Therefore, they cannot distinguish important parts of an article from stereotyped expressions such as *the paper describes* and *we find that.*

Our contributions To address these challenges, in this paper, we propose POLAR (PersOnaLized Article Recommendation framework), to combine the attention-based CNN with one-shot learning. Our main contributions can be summarized as follows:

1. We define the personalized article recommendation problem and show that it can be tackled in the framework of one-shot learning [14]. By transferring the method for classification to the ranking problem, we can overcome the sparsity of user feedback and improve the performance.
2. Based on the matching matrix in [20], we propose the attention matrix for text similarity, in which the importance of a term is calculated as the combination of the local and global weights.
3. Inspired by the success of convolutional neural network(CNN) [13] in image recognition, we build a CNN on the matching matrix and the attention matrix to capture the text similarity from word level to article level.

4. We conduct experiments on datasets of different sources and scales. Empirical results show that our framework can perform stably and significantly better than other comparative methods.

Organization The rest of the paper is organized as follows: Section 2 reviews related work. Section 3 is devoted to our POLAR framework. Section 4 presents experimental results and Section 5 concludes the paper.

2 Related Work

Personalized Recommendation Personalized recommender systems aim to recommend the most relevant items to a particular user in a given context. Content-based methods [21] compare item descriptions to the user profile to determine what to be recommend. Collaborative filtering methods [2] make rating prediction utilizing the past ratings of current user or similar users [16, 22], or the combination of these two [5]. To combine the advantages of the former two groups of methods, hybrid methods [15] are further proposed to improve the user profile modeling.

One-shot Learning One-shot learning is important for classification in cases where few examples are available. The method in [14]models the knowledge learned in other classes as a prior probability function w.r.t. the model parameters. Given an exemplar of a novel class, they update the knowledge and generate a posterior density to recognize novel instances. In [10], a Siamese network is learned with several convolutional layers used before the fully-connected layers and the top-level energy function. Matching Nets [28] take as input not only the new sample, but a small support set which contains labeled examples. Embedding functions are implemented by an LSTM with read-attention over the support set. While all these models perform on image tasks, we take a step further and propose a one-shot learning framework to recommend articles.

Text Similarity Traditional methods for measuring the similarity between two articles, such as BM25 [23] and TF-IDF [25], are based on the bag-of-words model. These methods often take as the similarity score the sum of weights of matched words in two articles. They don't perform well on identifying the matching of phrases and sentences.

Models based on neural networks can be categorized into two groups. The first group, called *representation based models*, get the distributed semantic representation of an article with neural networks and then take as the similarity score the similarity (often cosine similarity) between distributed representations of two articles. This group include DSSM [8],LSTM-RNN [19] and MV-LSTM [29]. However, these models often lack the ability to identify the specific matching signals. The second group of models, called *interaction based models*, use neural networks to learn the patterns in the word-level interaction of two articles, usually based on word embeddings, such as MatchPyramid [20] and K-NRM [31]. The DRMM [7] uses a multilayer perceptron over a histogram of word similarities to get the similarity score of two articles. These models lack the explicit

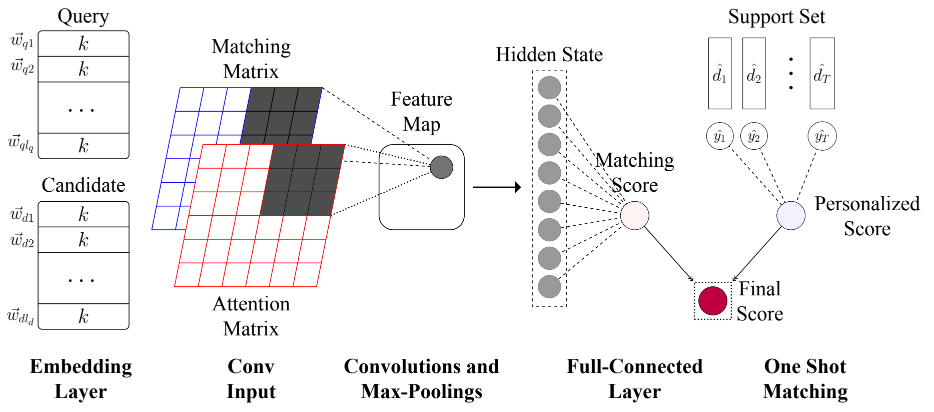

Fig. 1: The architecture of the overall framework. The articles are transformed into sequences of word embeddings through the embedding layer. The attention matrix and matching matrix are computed and sent to the CNN. The matching scores are combined with the support set to get the final scores.

expressions of word weights but rather depend on the characteristics of word embeddings. Representation based models and interaction based models have been combined in Duet [18] to improve the performance.

3 Approach

3.1 Framework with One-shot Learning

To get the ordered set R, for each article d_i in D our model computes a score $s(d_i|d_q, S)$, and k articles in D with the largest scores are selected as the top-k recommendation.

The recommendation problem for a specific user u can be considered as identifying whether u will accept an article or not and converted into binary classification. For each $(\hat{d}, \hat{y}) \in S$, $\hat{y}$ is binary(1 for relevant and 0 for irrelevant). S can be seen as the training set for classification, where $\hat{d}$ is a training instance and $\hat{y}$ is the corresponding label. It is probable to make an analogy between one-shot learning and our problem because S is of very limited size or even empty. Inspired by [28], our model computes $s(d_i|d_q, S)$ as follows:

$$s(d_i|d_q, S) = \begin{cases} \mathrm{c}(d_q, d_i) & S = \varnothing \\ \mathrm{c}(d_q, d_i) + \frac{1}{|S|} \sum\limits_{(\hat{d},\hat{y}) \in S} \mathrm{c}(\hat{d}, d_i)\hat{y} & S \neq \varnothing \end{cases} \tag{1}$$

where $c(\cdot, \cdot)$ is our attention-based CNN for text similarity, which will be discussed in the following part. The first part of s is the *matching score* with the query article. The second part, the *personalized score*, is the normalized linear combination of the feedback in S with text similarity as coefficients, and equals zero when S is empty. The whole framework is illustrated in Figure 1.

3.2 Matching Matrix and Attention Matrix

Each article d_i is a sequence of l_i terms $[t_{i1}, t_{i2}, \cdots, t_{il_i}]$ (We use *term* instead of *word* to show that the article has gone through preprocessing including tokenization and removal of stopwords). The matching matrix of article d_m and d_n, $\boldsymbol{M}^{(m,n)} \in \mathbb{R}^{l_m \times l_n}$, is defined as follows:

$$\boldsymbol{M}_{i,j}^{(m,n)} = \frac{\boldsymbol{w}_{mi}^T \cdot \boldsymbol{w}_{nj}}{\|\boldsymbol{w}_{mi}\| \cdot \|\boldsymbol{w}_{nj}\|} \tag{2}$$

where $\boldsymbol{w}_{mi}$ and $\boldsymbol{w}_{nj}$ are the word embeddings of term t_{mi} and t_{nj}. Since the cosine similarity of word embeddings can capture the semantic similarity [17], $\boldsymbol{M}_{i,j}^{(m,n)}$ is the similarity between t_{mi} and t_{nj}.

Since all terms are treated equally in the matching matrix without any weighting, the matching matrix cannot reflect the term importance. Therefore the matching matrix cannot distinguish the matching signals of important terms from those of structural, unimportant terms.

Table 1: Attention mechanisms in CNN

Method	Description
Object Parts Selection	In *fine-grained classification* [30], image patches which contain parts of certain objects are selected through a supervised process to extract discriminative features.
Attention Matrix	In [32], an attention matrix is employed to give different attention weights to units in a feature map.
Configurable Convolution	For *visual question answering* task [4], configurable convolutional kernels are generated by transforming the question embeddings from the semantic space into the visual space, which implements the question-guided attention.

To add the attention mechanism, we go over several applications of the attention mechanism in CNN in Table 1. We think the attention matrix, which can represent the importance of units in the feature map, quite suitable for our problem. The attention matrix, $\boldsymbol{A}^{(m,n)} \in \mathbb{R}^{l_m \times l_n}$ is defined as follows.

$$\boldsymbol{A}_{i,j}^{(m,n)} = r_{mi} \cdot r_{nj} \tag{3}$$

where r_{mi} and r_{nj} are the weights of term t_{mi} and t_{nj}. $\boldsymbol{M}^{(m,n)}$ and $\boldsymbol{A}^{(m,n)}$ are combined as the input of CNN.

3.3 Local Weight and Global Weight

Traditional methods for texts similarity often combine two types of term weights: the local weight, which depends on the specific document where the term occurs, and the global weight, which relies on the property of the whole corpus. Take the TF-IDF [25] method as an example. The TF (term frequency, how many times the term occurs in the given document) is the local weight and the Inverse DF (document frequency, how many documents the term occurs in) is the global weight.

We also combine the two weights in our model. The final weight of a term is the product of its local and global weights:

$$r_{ij} = \mu_{ij} \cdot \upsilon_{ij} \tag{4}$$

where μ_{ij} and υ_{ij} are respectively the local and global weights of the term t_{ij}.

Local Weight: How relevant is the term to the subject of the document? The local weight measures the relevance of a term to the subject of the document. For example, in the following text [6]:

Example 1. We propose a low-complexity audio-visual person authentication framework based on multiple features and multiple nearest-neighbor classifiers. The proposed MCCN method delivers a significant separation between the scores of client and impostors as observed on trials run on a unique database.

nearest-neighbor, *classifier* and *features* are obviously more important than *complexity* and *database*, and should have higher local weights, because they are more related to the topic of the text: *audio-visual authentication*.

Traditionally, the local weight is a math function of the frequency that the term occurs in a document, such as term frequency (TF) in TF-IDF [25] or the latter part in BM25 [23] ranking function:

$$\mathrm{BM25}(d, q) = \sum_{i=1}^{n} \mathrm{IDF}(q_i) \cdot \frac{\mathrm{TF}(q_i, d)(k_1 + 1)}{\mathrm{TF}(q_i, d) + k_1(1 - b + b\frac{|d|}{avgdl})} \tag{5}$$

where q_i is the i-th term of the query, $TF(q_i, d)$ is the term frequency of q_i in d and $avgdl$ is the average length of documents. k_1 and b are free parameters.

The basic idea of these methods is that the more important for a document a term is, the more frequently it occurs in the document. This is not always true. In Example 1, the important terms such as *authentication* and *classifier* occur only once, while the terms that occur more than once are stopwords like *of* and *on*. Therefore, a better mechanism for local weights is needed.

Local Weight Network Inspired by [34], we propose a local weight network based on distributed word representations. The basic idea is that, because of the linearity of word embeddings, the subject of a document can be expressed as the mean of vectors of its terms. The difference between the mean vector and term vector can be seen as the *semantic difference* between the document and the term.

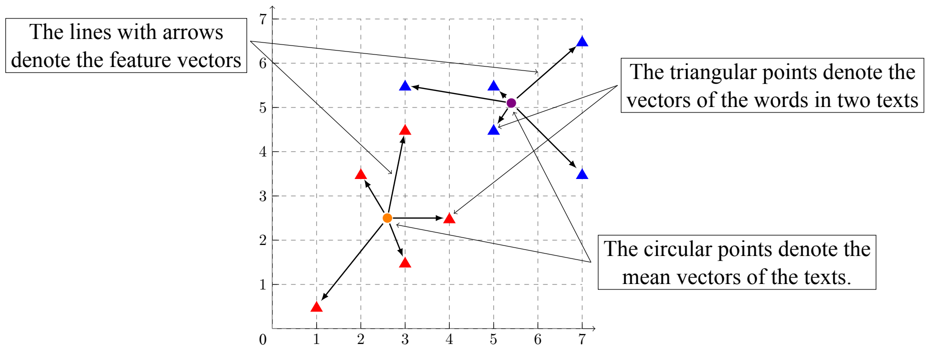
The lines with arrows denote the feature vectors
The triangular points denote the vectors of the words in two texts
The circular points denote the mean vectors of the texts.
0
1
2
3
4
5
6
7

Fig. 2: A two-dimensional example of the feature vectors for local weights

To compute the local weight μ_{ij}, the feature vector $\boldsymbol{x}_{ij}$ is the difference between the word vector $\boldsymbol{w}_{ij}$ and the mean vector of d_i:

$$\boldsymbol{x}_{ij} = \boldsymbol{w}_{ij} - \overline{\boldsymbol{w}}_i \tag{6}$$

where

$$\overline{\boldsymbol{w}}_i = \frac{1}{n_i}\sum_{k=1}^{n_i} \boldsymbol{w}_{ik} \tag{7}$$

Figure 2 gives an illustration of the feature vector.

We employ a feed forward network to learn the patterns in the feature vector $\boldsymbol{x}_{ij}$ and produce the local weight. The network is a multilayer perceptron(MLP) with multiple hidden layers and gives outputs within an interval.

$$\boldsymbol{u}_{ij}^{(0)} = \boldsymbol{x}_{ij} \tag{8}$$

$$\boldsymbol{u}_{ij}^{(l)} = \mathrm{ReLU}(\mathrm{BN}(\boldsymbol{W}^{(l-1)} \cdot \boldsymbol{u}_{ij}^{(l-1)} + \boldsymbol{b}^{(l-1)})), l = 1, 2, \cdots, L \tag{9}$$

$$\mu_{ij} = \sigma(\boldsymbol{W}^{(L)} \cdot \boldsymbol{u}_{ij}^{(L)} + \boldsymbol{b}^{(L)}) + \alpha \tag{10}$$

where L is the number of hidden layers in the feed forward network, BN is Batch Normalization [9] between the affine transformation and ReLU non-linearity and σ is the Sigmoid function. α is a nonnegative hyperparameter to set a lower bound and avoid giving a term a local weight close to 0. The ratio of maximum value to minimum value of local weights is $1 + \frac{1}{\alpha}$. It indicates that the smaller α is, the wider the range of local weights is.

Global Weight: How distinctive is the term in the whole corpus? The global weight measures how distinctive and specific a term is. It is independent of the specific document, but depends on the whole corpus. For example, in a set of papers on computer science, *computer* and *software* are less specific than *medicine* and *neural* and should be given lower global weights. But in a medical document corpus, it may just be the reverse.

The most widespread form of global weights is the inverse document frequency (IDF). The idea is that the specificity of a term can be quantified as an inverse function of its document frequency. There are a whole family of inverse functions, and the most common one is:

$$\mathrm{IDF}(t) = \log(\frac{N}{n_t}) \tag{11}$$

where t is the aim term, n_t is the document frequency of t and N is the total number of documents in the corpus.

Since the IDF measure has long been used and the use of other measures such as PageRank didn't lead to better results, here we also employ IDF as the measure of global weights. But to narrow the range of global weights and control the effect, instead of the raw IDF values, we use:

$$\upsilon_{ij} = [\mathrm{IDF}(t_{ij})]^{\beta} \tag{12}$$

where β is a hyperparameter within the interval (0,1). The smaller β is, the narrower the range of global weights is.

3.4 Convolutional Neural Network

The matching matrix and attention matrix are combined by element-wise multiplication and sent to a CNN, which consists of several convolutional layers and max-pooling layers:

$$\boldsymbol{Z}^{(0,0)} = \boldsymbol{M} \otimes \boldsymbol{A} \tag{13}$$

$$\boldsymbol{Z}_{x,y}^{(l+1,k')} = \mathrm{ReLU}(\mathrm{BN}(\sum_{k=0}^{c_l-1}\sum_{i=0}^{r_k-1}\sum_{j=0}^{r_k-1} \boldsymbol{w}_{i,j}^{(l+1,k)} \cdot z_{x+i,y+j}^{(l,k)} + b^{(l+1,k)})) \\ k' = 0, 1, \cdots, c_l, l = 0, 2, 4, \cdots, \tag{14}$$

$$\boldsymbol{Z}_{x,y}^{(l+1,k)} = \max_{0\le i<d_k} \max_{0\le j<d_k} z_{x\cdot d_k+i, y\cdot d_k+j}^{(l,k)}, l = 1, 3, 5, \cdots, \tag{15}$$

Similar to CNNs in image recognition [33], the filters in low-level convolutional layers can capture different matching signals between phrases, while the filters in high-level convolutional layers can capture the matching signals between sentences and paragraphs. The max-pooling layers can downsample the signals and reduce the spatial size of feature maps.

The output of the last max-pooling layer is then turned into a vector and passed through an MLP with several hidden layers, as described in Equation 9. In this paper, we use only one hidden layer. For the final output, a single unit is connected to all the units of the last hidden layer.

3.5 Optimization and Training

The entire model, including the CNN and the local weight network, is trained end-to-end on the target task.

The hinge loss is used as the objective function for training. Given the triples $\{(d_q^{(i)}, d_+^{(i)}, d_-^{(i)})\}_{i=1}^N$, where article $d_+^{(i)}$ is ranked higher than article $d_-^{(i)}$ with respect to query article $d_q^{(i)}$, the loss is:

$$Loss = \sum_{i=1}^{N} \max(0, 1 - c(d_q^{(i)}, d_+^{(i)}) + c(d_q^{(i)}, d_-^{(i)})) \tag{16}$$

where $c(d_q, d)$ denotes the predicted matching score between d_q and d.

Since the size of some datasets we use is relatively small, for experiments on these datasets we train the model on a classifying task called citation prediction. Given the abstracts of two papers, the model needs to classify them as having citation relationship or not. Obviously, to complete the task, the model also needs to compute the relevance of two articles. In this case the loss function is the cross entropy. Given the triples $\{(d_1^{(i)}, d_2^{(i)}, y^{(i)}\}_{i=1}^N$, the loss is:

$$\begin{aligned} Loss &= -\sum_{i=1}^{N} y^{(i)} \log(p^{(i)}) + (1 - y^{(i)}) \log(1 - p^{(i)}) \\ p^{(i)} &= \frac{1}{1 + \mathrm{e}^{-c(d_1^{(i)}, d_2^{(i)})}} \end{aligned} \tag{17}$$

where $p^{(i)}$ is the predicted probability that the i-th instance is positive and $y^{(i)}$ is the label of the i-th instance.

The optimization is done through standard backpropagation [24] and stochastic gradient descent method with mini-batches. For regularization, we use dropout [26] in the output of every hidden layer and early stopping strategy [3] to avoid over-fitting.

4 Experiments

In this section, to evaluate the proposed model, we conduct experiments on the article recommendation problem based on three datasets, in comparison with traditional methods and neural models.

4.1 Experiment Setup

Comparison Methods The following are several traditional methods.

- **TF-IDF [25]**: The similarity score between a query and a document is computed by summing the weights of the query's terms which also occur in the document. The weight of a term is the product of its TF and IDF weights.

– **Doc2Vec [12]**: We get the distributed representation of each article via Paragraph Vector model. The similarity score between two articles is produced by the cosine similarity of their representations.
– **WMD [11]**:The Word Mover's Distance (WMD) is the minimum distance required to transport words from one document to another based on the word embeddings.

The following are several neural matching models.

– **MV-LSTM [29]**:The interactions between different positional sentence representations generated by a Bi-LSTM form a similarity matrix to generate the matching score.
– **MatchPyramid [20]**: A CNN is built on the standard matching matrix to get the matching score.
– **DRMM [7]**:The matching between the terms in the query and the document is expressed as a histogram, where only the counts of the matching score in different intervals are reserved. The histogram is sent to an MLP to get the matching score.
– **Duet [18]**:An interaction-based model and a representation-based model are combined to get the matching score of two articles.

Parameter Setting In the Local Weight Network there are two hidden layers, with 64 and 32 hidden units respectively. The CNN has three convolutional layers and three max-pooling layers. The first and second convolutional layers both have 32 filters and the third convolutional layer has 16 filters. All convolutional filters are set to 3×3 and all max-pooling kernels are set to 2×2. The number of hidden units in the full-connected layer is set to 256. For the hyperparameters α and β, we set $\alpha = 1$ and $\beta = \frac{1}{4}$, which is discussed in Section 4.3.

The word embeddings in all the models above are 256 dimensions trained on Wikipedia via the skip-gram model, using hierarchical softmax and negative sampling [17].

Dataset We evaluate the performance of the proposed model with two small, manually labeled datasets and a large-scale dataset based on user click.

The first dataset is based on papers from AMiner [27] and consists of 188 query papers with 10 candidate papers for each query. The second dataset is based on documents of patents coming from the Patent Full-Text Databases of the United States Patent and Trademark Office[2] and consists of 67 queries with 20 candidates for each query. In each dataset, we gather relevance judgments from college students or experts on patent analysis as the ground truth. The relevance is simply expressed as binary: relevant or irrelevant. Abstracts of the papers or the patent documents are used as texts and texts longer than 96 terms are truncated.

[2] `http://patft.uspto.gov/`

Table 2: Results of relevance ranking(%). NG stands for NDCG

Method	**AMiner**			**Patent**			**RARD**		
	NG@3	NG@5	NG@10	NG@3	NG@5	NG@10	NG@1	NG@3	NG@5
TF-IDF	74.3	81.8	87.5	51.8	56.4	63.4	37.6	39.8	46.3
Doc2Vec	60.0	65.8	79.1	44.6	45.6	53.5	28.4	34.0	40.0
WMD	73.0	76.3	86.2	57.4	58.5	61.9	23.4	38.2	46.8
MV-LSTM	56.2	61.2	76.2	60.2	59.0	65.0	22.2	30.7	39.3
Duet	66.6	74.4	82.6	54.5	57.5	64.6	22.3	31.1	39.8
DRMM	75.0	79.9	87.1	55.0	56.2	64.7	33.1	36.3	40.6
MatchPyramid	73.5	80.0	86.8	56.4	61.4	64.4	29.1	36.2	42.8
POLAR	**80.3**	**85.2**	**90.1**	**67.8**	**69.5**	**73.6**	**42.8**	**46.3**	**51.5**

Since the sizes of two datasets are relatively small, we train the models on the citation prediction task, which is described in Section 3.5, for all comparison methods. The dataset for training is the Citation Network Dataset in AMiner [27].

The third dataset is Related-Article Recommendation Dataset(RARD) [1] from Sowiport, a digital library of social science articles that displays related articles to its users. The dataset contains 63923 distinct queries with user click log. Each query article has an average of 9.1 articles displayed. The displayed documents are generated by a recommender-as-a-service provider Mr. DLib, so they are of high relevance to the query. We choose 800 queries that have the most clicks for test and other queries are used for training. Since the abstracts of some articles are missing, the titles and the abstracts of articles are combined as texts. Texts longer than 64 terms are truncated.

4.2 Performance Comparison

Table 2 shows the ranking accuracy of different methods in terms of NDCG. For the fairness of comparison, all models don't involve user feedback, which will be discussed in Section 4.3.

From the evaluation results, we can observe that our proposed model POLAR can perform better than all the baselines. POLAR can outperform the best baselines 6.9%-13.2% on NDCG@3 and 3.3%-20.3% on NDCG@5. The average improvements of NDCG on each dataset are respectively 3.8%, 8.1% and 6.4%.

Among the traditional ranking models, TF-IDF is the most competitive one, in some cases even outperforming the best neural baselines by 5.5%. But we can also find that TF-IDF performs not very well on the patent dataset. The reason might be that documents of patents are often written by non-academic researchers and terms on the same topic might vary from person to person. Only taking the exact matching signals into account, TF-IDF might be unsuitable for such situation, while the methods based on word embeddings can perform better.

As for the neural ranking models, we can see that interaction based models, including DRMM and MatchPyramid, perform slightly better than representation based models. Although the Duet combines the interaction-based model and the representation-based model, it doesn't perform better than individual interaction-based models.

4.3 Analysis and Discussion

How one-shot learning can help We utilize the datasets in the previous part to simulate the personalization problem. We select those queries that have more than one positive-labeled candidate. For every query, we randomly divide the labeled documents into two parts. The first part is used as the support set and the second part is used as the candidate set to recommend. Then we compare the proposed one shot framework (called POLAR-OS) with the best model that ignores support sets in the previous part (called POLAR-ALL). The support set is quite sparse compared with the size of candidates. For example, in the RARD dataset, the average size of support set for each query is only 1.5. In the AMiner dataset, the size of support set is only 1 for 45% queries and 2 for 47%. In the patent dataset, the sizes of support sets of 75% queries are no greater than 3.

The result is shown in table 3. We can see that the performance can be improved with a small amount of feedback data. On average, POLAR-OS can outperform POLAR-ALL by 7.0% on NDCG@1 and 5.7% on NDCG@3.

Table 3: Performance for the model with one shot learning and without.

	AMiner		**Patent**		**RARD**	
Method	NDCG@1	NDCG@3	NDCG@1	NDCG@3	NDCG@1	NDCG@3
POLAR-ALL	76.1	79.2	52.3	66.2	36.5	36.5
POLAR-OS	**79.1**	**81.9**	**57.1**	**69.7**	**39.4**	**39.2**

How the attention matrix can help To illustrate the improvements different parts of the attention matrix bring, we compare three versions of the proposed model with different attention matrices. To compute the attention matrix, POLAR-LOC uses only the local weights and POLAR-GLO uses only the global weights. POLAR-ALL uses both local weights and global weights. The performance in terms of NDCG@3 is shown in Figure 3.

In most cases, POLAR-LOC, the model with the local weight network, performs better than POLAR-GLO. The reason might be that the local weight network is trainable, with greater ability to learn the importance of terms. IDF is only a statistical way to get approximate values. The complete model, POLAR-ALL, which combines the two weights, performs significantly better than either

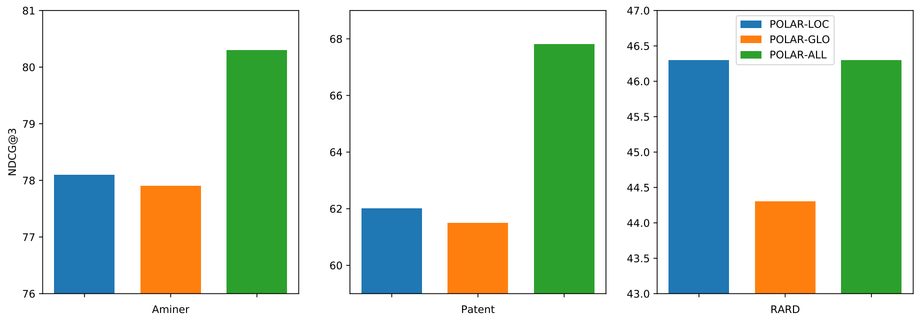

Fig. 3: The performance of different attention matrices

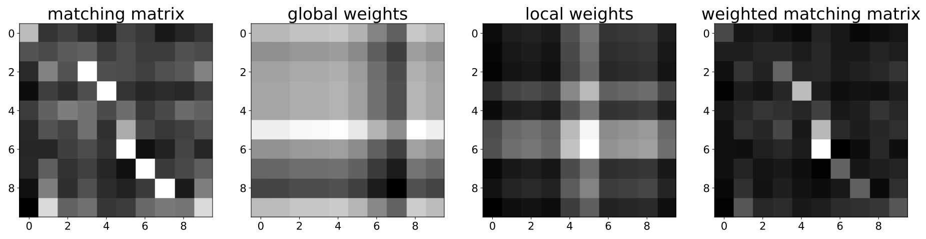

Fig. 4: The visualization result of four matrices used in the matching of a pair of texts. The brighter the pixel is, the larger value it has. The text pair is as follows(the words in brackets are removed stopwords):
T1:novel robust stability criteria (for) stochastic hopfield neural networks (with) time delays.
T2:new delay dependent stability criteria (for) neural networks (with) time varying delay.

of them. This confirms that the local and global weights are complementary to each other.

To have a better understanding of how local and global weights work, we show the pixel images of four matrices in Figure 4. From the images we can find that the local weights of most terms are low while the global weights of most terms are high. The statistical analysis of the local and global weights in Table 4 also supports this idea. Therefore, we can conclude that the global weights function by deemphasizing unimportant terms in the corpus with low weights, while the local weights function by highlighting key terms in specific articles.

Sensitivity Analysis of Hyperparameters Since there are two hyperparameters α and β to control the effect of the local and global weights in our proposed model, we further study the effect of different choices of α and β. The result is shown in figure 5 In general, the variance in β has greater effect than that in α. In our model the global weights are predefined values which couldn't be changed

Table 4: The statistical analysis of the local and global weights

Weight	Max	Min	Mean	Std
Local	2.00	1.00	1.20	0.15
Global	1.96	1.08	1.86	0.08

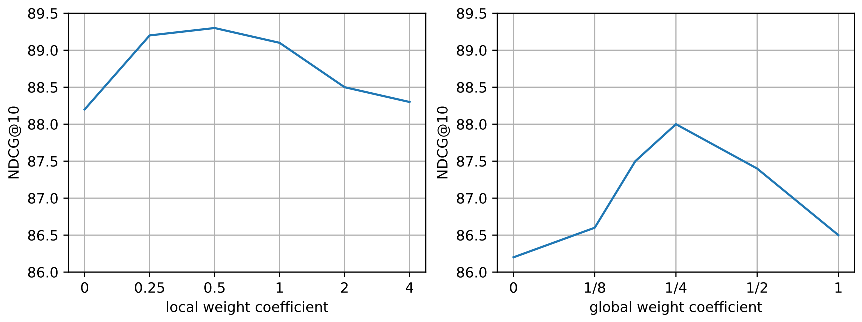

Fig. 5: Performance comparison for POLAR-LOC with different α and POLAR-GLO with different β on the AMiner dataset

once β is chosen, while the local weights are calculated by the local weight network, which can automatically adapt to different choices of α. Therefore it is important to choose the value of β. When β is close to 1, the global weights are equal to IDF values, which vary so greatly that the model will ignore the effect of cosine similarity. When β is close to 0, the global weights are almost uniform and have little effect.But the model with the value of α equal to 0 cannot perform well either, because the local weight network can have too strong effect and be troubled by over-fitting.

5 Conclusion

In this paper, we study the problem of personalized article recommendation. We define the problem and propose a novel model POLAR to solve it. We utilize the framework of one shot learning to deal with the sparse user feedback and propose an attention based CNN model for text similarity. Experimental results show that the proposed model significantly outperforms both the traditional and the state-of-art neural baselines. The model has been used in AMiner to provide recommendation of similar papers.

For further work, we would like to combine our model with reinforcement learning (RL), to train a deeper and more powerful model in the online environment. We may also compare the performance of different attention mechanisms in CNN.

References

1. Beel, J., Carevic, Z., Schaible, J., Neusch, G.: Rard: The related-article recommendation dataset [data] (2017). https://doi.org/10.7910/DVN/HA8EAH
2. Breese, J.S., Heckerman, D., Kadie, C.: Empirical analysis of predictive algorithms for collaborative filtering. In: UAI. pp. 43–52 (1998)
3. Caruana, R., Lawrence, S., Giles, L.: Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping. In: NIPS. pp. 381–387 (2000)
4. Chen, K., Wang, J., Chen, L., Gao, H., Xu, W., Nevatia, R.: ABC-CNN: an attention based convolutional neural network for visual question answering. CoRR **abs/1511.05960** (2015)
5. Das, A.S., Datar, M., Garg, A., Rajaram, S.: Google news personalization: Scalable online collaborative filtering. In: WWW. pp. 271–280 (2007). https://doi.org/10.1145/1242572.1242610
6. Das, A.: Audio visual person authentication by multiple nearest neighbor classifiers. In: ICB. pp. 1114–1123 (2007). https://doi.org/10.1007/978-3-540-74549-5_116
7. Guo, J., Fan, Y., Ai, Q., Croft, W.B.: A deep relevance matching model for ad-hoc retrieval. In: CIKM. pp. 55–64 (2016). https://doi.org/10.1145/2983323.2983769
8. 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: CIKM. pp. 2333–2338 (2013). https://doi.org/10.1145/2505515.2505665
9. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: ICML. pp. 448–456 (2015)
10. Koch, G., Zemel, R., Salakhutdinov, R.: Siamese neural networks for one-shot image recognition. In: ICML Deep Learning Workshop. vol. 2 (2015)
11. Kusner, M.J., Sun, Y., Kolkin, N.I., Weinberger, K.Q.: From word embeddings to document distances. In: ICML. pp. 957–966 (2015)
12. Le, Q.V., Mikolov, T.: Distributed representations of sentences and documents. In: ICML. pp. 1188–1196 (2014)
13. Lecun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86**(11), 2278–2324 (1998). https://doi.org/10.1109/5.726791
14. Li, F., Fergus, R., Perona, P.: One-shot learning of object categories. IEEE Trans. Pattern Anal. Mach. Intell. **28**(4), 594–611 (2006). https://doi.org/10.1109/TPAMI.2006.79
15. Li, L., Wang, D., Li, T., Knox, D., Padmanabhan, B.: Scene: A scalable two-stage personalized news recommendation system. In: SIGIR. pp. 125–134 (2011). https://doi.org/10.1145/2009916.2009937
16. Marlin, B., Zemel, R.S.: The multiple multiplicative factor model for collaborative filtering. In: ICML. pp. 73– (2004). https://doi.org/10.1145/1015330.1015437
17. Mikolov, T., Sutskever, I., Chen, K., Corrado, G., Dean, J.: Distributed representations of words and phrases and their compositionality. In: NIPS. pp. 3111–3119 (2013)
18. Mitra, B., Diaz, F., Craswell, N.: Learning to match using local and distributed representations of text for web search. In: WWW. pp. 1291–1299 (2017). https://doi.org/10.1145/3038912.3052579
19. Palangi, H., Deng, L., Shen, Y., Gao, J., He, X., Chen, J., Song, X., Ward, R.: Deep sentence embedding using long short-term memory networks: Analysis and application to information retrieval. IEEE/ACM Trans. Audio, Speech and Lang. Proc. **24**(4), 694–707 (2016). https://doi.org/10.1109/TASLP.2016.2520371

20. Pang, L., Lan, Y., Guo, J., Xu, J., Wan, S., Cheng, X.: Text matching as image recognition. In: AAAI. pp. 2793–2799 (2016)
21. Pazzani, M.J., Billsus, D.: Content-based recommendation systems. In: The Adaptive Web, Methods and Strategies of Web Personalization. pp. 325–341 (2007). https://doi.org/10.1007/978-3-540-72079-9_10
22. Rendle, S.: Factorization machines. In: ICDM. pp. 995–1000 (2010). https://doi.org/10.1109/ICDM.2010.127
23. Robertson, S.E., Walker, S., Jones, S., Hancock-Beaulieu, M., Gatford, M.: Okapi at TREC-3. In: TREC. pp. 109–126 (1994)
24. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning representations by back-propagating errors. Nature **323**(6088), 533–536 (1986)
25. Salton, G., Fox, E.A., Wu, H.: Extended boolean information retrieval. Commun. ACM **26**(11), 1022–1036 (1983). https://doi.org/10.1145/182.358466
26. Srivastava, N., Hinton, G.E., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. JMLR **15**(1), 1929–1958 (2014)
27. Tang, J., Zhang, J., Yao, L., Li, J., Zhang, L., Su, Z.: Arnetminer: Extraction and mining of academic social networks. In: SIGKDD. pp. 990–998 (2008). https://doi.org/10.1145/1401890.1402008
28. Vinyals, O., Blundell, C., Lillicrap, T., Kavukcuoglu, K., Wierstra, D.: Matching networks for one shot learning. In: NIPS. pp. 3630–3638 (2016)
29. Wan, S., Lan, Y., Guo, J., Xu, J., Pang, L., Cheng, X.: A deep architecture for semantic matching with multiple positional sentence representations. In: AAAI. pp. 2835–2841 (2016)
30. Xiao, T., Xu, Y., Yang, K., Zhang, J., Peng, Y., Zhang, Z.: The application of two-level attention models in deep convolutional neural network for fine-grained image classification. In: CVPR. pp. 842–850 (2015). https://doi.org/10.1109/CVPR.2015.7298685
31. Xiong, C., Dai, Z., Callan, J., Liu, Z., Power, R.: End-to-end neural ad-hoc ranking with kernel pooling. In: SIGIR. pp. 55–64 (2017). https://doi.org/10.1145/3077136.3080809
32. Yin, W., Schütze, H., Xiang, B., Zhou, B.: ABCNN: attention-based convolutional neural network for modeling sentence pairs. TACL **4**, 259–272 (2016)
33. Zeiler, M.D., Fergus, R.: Visualizing and understanding convolutional networks. In: ECCV. pp. 818–833 (2014). https://doi.org/10.1007/978-3-319-10590-1_53
34. Zheng, G., Callan, J.: Learning to reweight terms with distributed representations. In: SIGIR. pp. 575–584 (2015). https://doi.org/10.1145/2766462.2767700