POLAR++: Active One-Shot Personalized Article Recommendation

Zhengxiao Du, Jie Tang, and Yuhui Ding

Abstract—We study the problem of personalized article recommendation, in particular when the user's preference data is missing or limited, which is knowns as the user cold-start issue in recommender systems. We propose POLAR++, an active recommendation framework that utilizes Bayesian neural networks to capture the uncertainty of user preference, actively selects articles to query the user for feedback, and adaptively learns user preference with one-shot learning. For the article recommendation, we design an attention-based CNN to quantify the similarity between user preference and recommended articles, which significantly improves the performance with only a few articles rated by the users. We evaluate the proposed POLAR++ on datasets of different scale and sources. Experimental results demonstrate the effectiveness of the proposed model. We have successfully deployed POLAR++ into AMiner as the recommendation engine for article recommendation, which further confirms the effectiveness of the proposed model.

Index Terms—Recommender systems, active learning, one-shot learning, cold-start

1 Introduction

Recommender systems play a key role in today's web applications. For example, in academic search sites such as Google Scholar and AMiner [1], article recommendation is essential for users to find the right articles as the number of articles has been increasing dramatically in the past years. The recently released Open Academic Graph (OAG)[1] consists of 208,915,369 papers, 52,678 venues, and 253,144,301 authors [2]. Many digital library providers article recommendations to help users find recent or related articles. These recommendations are often based on keyword similarity between the current article and candidate articles.

An article may cover several different topics. For example, this current paper covers *recommender systems, active learning, one-shot learning, and cold-start*. Users with different backgrounds and interests may be interested in different topics. Recommendation results without personalization may ignore user preference and diversity, thus cannot satisfy users.

Precisely capturing users' preferences is always challenging. Still taking the academic search as an example, a large portion of the users have the cold-start problem on various topics, partially because their profiles are incomplete or missing and partially because we cannot collect sufficient data for them on the different topics. Methods based on implicit user feedback are typically preferred [3]. However, the amount of user feedback might be limited. For new users, only real-time implicit feedback is possible. Therefore, it is difficult to directly apply the traditional recommendation methods such as Content-based Recommendation [4] or Collaborative Filtering [5] in this scenario. Inspired by the recent success of one-shot deep learning [6], [7], [8], we propose to learn a one-shot deep matching metric for personalized article recommendation by actively querying the user for feedback. For new users, we may not know their preferences. How can we *efficiently* and *effectively* acquire users' preferences through a few user interactions?

Motivating Example. Fig. 1 gives an example from AMiner.org to illustrate how to actively learn the preferences of a new user and recommending related articles. The left of the figure gives the initial state of our problem. A new user comes to the website and visits an article. Since we do not know the user's preferences, we cannot provide a personalized recommendation. The *Active Interaction* part shows our solution to this problem: we ask the user to give feedback to a few articles. Different colors represent articles in different fields. Once the user gives the feedback, with the help of one-shot learning, the model learns the user's preferences and accordingly recommends articles the user might be interested in, as shown in the right figure.

The problem now is the selection of the queried articles would greatly influence the final recommendation performance, as only with high-quality user feedback, one-shot learning can learn the user's preferences effectively. The articles with feedback are only a small part of articles the user might be interested in. As a result, the learned model might be overfitting to the user's interest with the limited feedback, covering only a small part of the user's interest. To this end, we propose to combine an active learning strategy with one-shot learning. However, it is still an open question on how to design a strategy to actively learn the user's preferences and intentions with minimal user efforts.

Another challenge with personalized article recommendation is that new articles come every day, without enough

1. https://www.openacademic.ai/oag/

• The authors are with the Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China.
E-mail: {duzx16, dingyh15}@mails.tsinghua.edu.cn, jietang@tsinghua.edu.cn.

Manuscript received 19 Nov. 2018; revised 8 Sept. 2019; accepted 29 Oct. 2019. Date of publication 15 Nov. 2019; date of current version 29 Apr. 2021.
(Corresponding author: Jie Tang.)
Recommended for acceptance by D. Cai.
Digital Object Identifier no. 10.1109/TKDE.2019.2953721

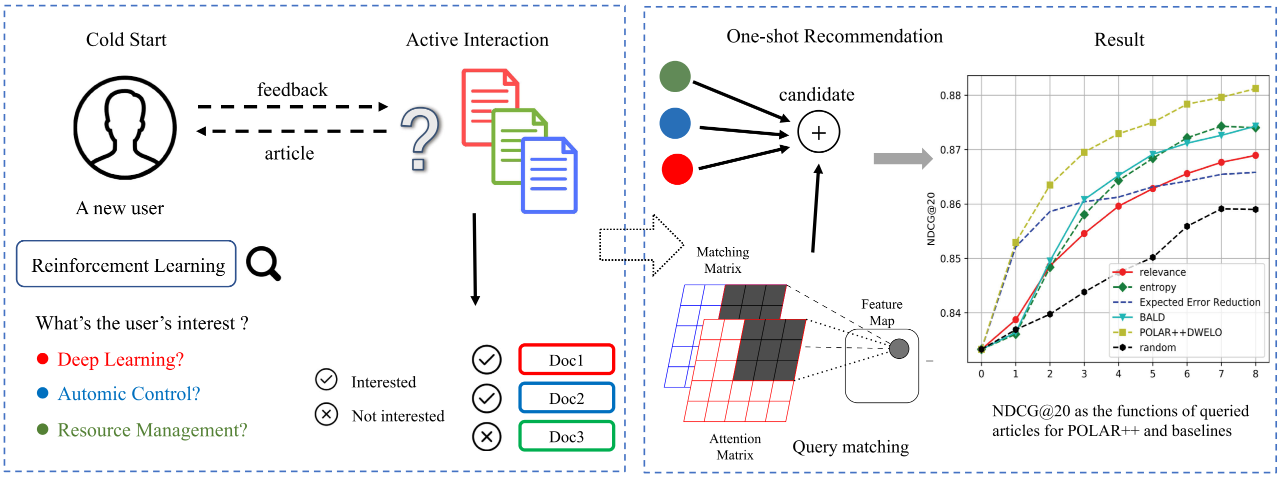


Fig. 1. Left: A motivation example for Active One-shot Article Recommendation Problem and our solution. Right: The one-shot personalized recommendation with the user's feedback and the ranking performance comparison of POLAR++ and several active learning baselines.

time to collect user opinions. Therefore, content-based recommendation methods are preferred [9], [10]. Articles often contain highly representative texts, like the abstract of a paper. Text similarity, which plays a crucial role in recommender systems and information retrieval, poses another challenge. The bag-of-words model, on which most traditional methods are based, ignores the information about word order and co-occurrence. Therefore these methods cannot capture the matching signals in phrases 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 [11], [12], [13], [14], [15], but they often treat all the words in an article indiscriminately. Therefore, they cannot distinguish essential parts of an article from stereotyped expressions such as *the paper describes* and *we find that*.

Contribution. To address these challenges, we propose POLAR++ (PersOnaLized Article Recommendation framework++),[2] to actively learn to provide personalized article recommendations. Our main contributions can be summarized as follows:

- To tackle the active one-shot article recommendation problem, we propose a recommendation framework based on Bayesian neural networks to capture the uncertainty of user preference and actively learn a new user's preference via user interactions. The Bayesian active learning method can be applied to any deep models for ranking or recommendation with a pairwise approach.
- Combining the proposed model with density-weighted Expected Loss Optimization [17], we introduce active learning into POLAR [16], an attention-based CNN combined with one-shot learning for personalized article recommendation to utilize extremely sparse implicit user feedback.
- We conduct experiments on datasets of different sources and scales. Empirical results show that our framework can perform stably and significantly better than comparative methods.

2. A prior version was published in [16].

Organization. The rest of the paper is organized as follows: Section 2 reviews related work. Sections 3 and 4 are devoted to our POLAR++ framework. The experimental setting is presented in Section 5 and the experimental results are analyzed in Section 6. Section 7 concludes the paper.

2 Related Work

Related literature of this work can be categorized into three groups: article recommendation, one-shot learning, and active learning. In this section, we briefly review literature in the three aspects.

2.1 Article Recommendation

Article recommendation plays an important role in academic search sites and digital libraries and has attracted a lot of research interest. Giles *et al*. introduced the first research-article recommender as part of the *CiteSeer* project [18]. Content-based filtering [4] is one of the most widely used and researched recommendation method and has been successfully applied in article recommendation [9], [19], [20], [21]. Most approaches use plain words as features, although some use n-grams [20], topics [9], [21], and citations [18]. Collaborative filtering [5] makes recommendation predictions by utilizing the explicit or implicit ratings of the current user and similar users [22]. However, in article recommendation, collaborative filtering often suffers from the cold-start problem [10]. Some works also use graph-based methods to explore the inherent connections in academia [23], [24].

Our work is also related to information retrieval [11], [25], [26] and semantic matching [12], [13]. Traditional methods for measuring the similarity between two pieces of texts, such as BM25 [26] and TF-IDF [25], are based on the bag-of-words model and do not perform well in 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 [11], [13], [27]. However, these models cannot 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 [12], [15], [28]. These models lack the explicit 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 [14] to improve the performance.

2.2 One-Shot Learning

One-shot learning is essential for classification in cases where only a few examples are available. The Bayesian method in [29] models the knowledge learned in other classes as a prior probability function w.r.t. the model parameters and generates a posterior density to recognize new instances given an exemplar of a novel class. Recent one-shot learning methods based on deep learning fall into two categories. *Metric-based approaches* try to learn a similarity metric to help predict the label of instance. In [6], a Siamese network is learned with several convolutional layers used before the fully-connected layers and the top-level energy function. Matching Nets [8] take as input not only the new sample but also a small support set that contains labeled examples. An LSTM with read-attention over the support set implements the embedding function. *Meta-learning-based approaches* aim to learn how to update the parameters or directly predict the parameters given a few training instances, including Memory-Augmented Network [7] and LSTM-based meta-learner [30].

2.3 Active Learning

Active learning is a subfield of machine learning in which a learning algorithm can choose the data from which it learns. It is closely related to Optimal Experiment Design [31] in the statistic literature. There are three different scenarios: membership query synthesis [32], stream-based sampling [33], and pool-based sampling [34]. One of the most common frameworks for active learning is *Uncertainty Sampling* [34], where the active learner selects the instance for which the prediction uncertainty is highest. The uncertainty measure includes Entropy [35] for classification and Variance [36] for regression. The drawback of uncertainty sampling is that it often samples the outliers or the instances with greater noise. *Query-by-Committee* [37] is more theoretically-motivated, which maintains a committee of models and minimizes the version space by querying in controversial regions of the input space, which are instances the committee disagree about most. Based on the decision theory, *Expected Error Reduction* [38], [39] aims to maximize the expected reduction of the generalization error, but is also the most computationally expensive framework.

Bayesian-based active learning has received much attention recently. A Bayesian information-theoretic active learning approach is presented in [40]. Unlabeled instances whose prediction the parameters under the posterior disagree about are selected. Expected Loss Optimization [17] selects the instance that maximizes the expected loss based on Bayesian decision theory. In [41] a Bayesian active learning algorithm for deep learning in image data is proposed based on the idea in [42].

Most works that apply active learning to recommender systems are based on collaborative filtering [43], [44], [45]. The active learning method is also called the Ask-To-Rate technique [46]. A comprehensive survey can be found in [47]. The Popularity strategy and the Coverage strategy are two representative heuristic methods, but they are not personalized. More advanced methods are based on uncertainty reduction [48] or error reduction [49], [50]. For example, in [44] a decision tree is built to model the Ask-To-Rate process for cold-start users. However, these methods are based on collaborative filtering methods with a fixed item set, while our method utilizes the text information to provide recommendations of the latest articles. Recently, an attribute-driven active learning method for item cold-start problem is proposed in [51]. They propose four heuristic criteria to select diverse and representative users for ratings. However, none of the four criteria considers the uncertainty of preference predictions, which is necessary for the efficiency in the user cold-start problem.

3 Preliminary

3.1 Problem Definition

To begin with, we first define the one-shot personalized article recommendation problem as follows.

Definition 1 (One-shot Personalized Article Recommendation Problem [16]). *The input of the problem is a query article d_q, the set of candidate articles $\mathcal{D} = \{d_i\}_{i=1}^{N}$, 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 $R(d_q, S) \subset \mathcal{D}$ with $|R| = k$, which is the top-k recommendation for u with respect to d_q.*

Note that either the query article d_q or the support set S could be empty (but not both of them). In the former case, it is the purely personalized article recommendation. In the latter case, it is the non-personalized related article recommendation. At this point we assume that the support set for a user u is fixed. For a new user, the support set is empty, which means that it is impossible to provide personalized recommendations for the user.

Moving from this, we define the new problem in the *pool-based sampling* setting [34] of active learning :

Definition 2 (Active One-shot Article Recommendation Problem). *The input of the problem is a query article d_q, an unlabeled set $\mathcal{U}$, and an interview budget b (number of feedback acquisition from the user). An active strategy π selects an article from $\mathcal{U}$ and gets its feedback from the user at each step until b article-feedback pairs are collected and form the actively-built support set Q. The output is the recommendation result $R(d_q, Q)$.*

We define the active learning problem in the *adaptive setting* (which is also called Personalized Active Learning in [47]). When the model selects the ith article, it has access to the previous $(i-1)$ articles and ratings, so that the strategy can dynamically adapt to different users and improve the efficiency of the interview process.

3.2 Bayesian Neural Networks

A Bayesian neural network is a neural network with a prior distribution on its parameters. Given the weight matrix $\mathbf{W}_i$

and the bias vector $\mathbf{b}_i$ for the ith layer, we often place a Gaussian distribution over the weight matrix:

$$p(\mathbf{W}_i) \sim \mathcal{N}(0, \sigma_i^2 \mathbf{I}), \tag{1}$$

For simplicity, we assume a point estimate for the bias vector $\mathbf{b}_i$.

Let $\boldsymbol{\omega} = \{\mathbf{W}_i\}_{i=1}^{L}$ denote the set of model parameters and $f^{\boldsymbol{\omega}}(\mathbf{x})$ the network output with respect to input $\mathbf{x}$ and parameter $\boldsymbol{\omega}$. Given a training set $\mathbb{D}_{train}$, the parameter posterior is $p(\boldsymbol{\omega}|\mathbb{D}_{train})$.

3.3 Variational Inference by Dropout

In practice, evaluation of the true posterior $p(\boldsymbol{\omega}|\mathbb{D}_{train})$ cannot be done analytically and an approximation is needed. We define an approximating variational distribution $q(\boldsymbol{\omega})$, which is easy to evaluate. The approximation should be as close to the posterior as possible, with minimal Kullback-Leibler(KL) divergence to the true posterior. Minimizing the KL divergence is equivalent to maximizing the *log evidence lower bound*:

$$\mathcal{L}_{VI} = \int q(\boldsymbol{\omega}) \log p(\mathbb{D}_{train}|f^{\boldsymbol{\omega}}(\mathbf{x}))\mathrm{d}\boldsymbol{\omega} - \mathrm{KL}(q(\boldsymbol{\omega})||p(\boldsymbol{\omega})), \tag{2}$$

which is the basic equation of Variational Inference [52].

Following [42], we use the distribution of the network parameter with dropout [53] as $q(\boldsymbol{\omega})$. Consider a neural network with only one layer, which gives output as $\mathbf{y} = \sigma(\mathbf{M}\mathbf{x} + \mathbf{b})$. If we use dropout with zeroing probability p on the input vector $\mathbf{x}$, , it's equivalent to

$$\begin{aligned}
\tilde{\mathbf{y}} &= \sigma(\mathbf{M}\tilde{\mathbf{x}} + \mathbf{b}) \\
&= \sigma(\mathbf{M}(\boldsymbol{\epsilon} \odot \mathbf{x}) + \mathbf{b}) \\
&= \sigma(\mathbf{M}(\mathrm{diag}(\boldsymbol{\epsilon}) \cdot \mathbf{x}) + \mathbf{b}) \\
&= \sigma((\mathbf{M} \cdot \mathrm{diag}(\boldsymbol{\epsilon}))\mathbf{x} + \mathbf{b}) \\
&= \sigma(\mathbf{W}\mathbf{x} + \mathbf{b}) \\
\epsilon_i &\sim \mathrm{Bernoulli}(1-p) \text{ for } i = 1, 2, \ldots, K.
\end{aligned}$$

Therefore dropout on $\mathbf{x}$ turns the weight matrix of the network into a random variable $\mathbf{W} = \mathbf{M} \cdot \mathrm{diag}(\boldsymbol{\epsilon})$ and $\mathbf{M}$ is the parameter of its distribution. We can apply dropout in every layer of a network. The corresponding distribution, which we call *dropout distribution*, can function as the approximating distribution $q(\boldsymbol{\omega})$.

4 APPROACH

Given a query document d_q and a support set S, for each article d_i in D, we predict the corresponding score $y_i = s(d_i|d_q, S)$. k articles in D with the largest recommendation scores are selected as the top-k recommendation.

To solve the Active One-shot Article Recommendation Problem, we model the uncertainty of recommendation scores explicitly in our proposed model. Therefore, unlike our previous work, the prediction of the recommendation score is a distribution, not a single point, of $\mathbb{R}$. We learn a Bayesian NN f that takes the triple $\mathbf{x}_i = (d_q, S, d_i)$ as input, and gives output $f^{\boldsymbol{\omega}}(\mathbf{x}_i)$. We define a likelihood function over the network's output and get the distribution $p(y_i|f^{\boldsymbol{\omega}}(\mathbf{x}))$.

It's noted that the model parameters $\boldsymbol{\omega}$ are random variables with prior and posterior distribution described in Section 3.2. With the posterior of parameters, we can get the predicted distribution given a new data point $\mathbf{x}$ by integrating:

$$p(y|\mathbf{x}, \mathbb{D}_{train}) = \int p(y|f^{\boldsymbol{\omega}}(\mathbf{x}))p(\boldsymbol{\omega}|\mathbb{D}_{train})\mathrm{d}\boldsymbol{\omega}, \tag{3}$$

which is also referred to as marginalizing the likelihood over $\boldsymbol{\omega}$.

4.1 One-Shot Personalized Recommendation

4.1.1 *One-Shot Personalization*

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 pair $(\hat{d}, \hat{y}) \in S$, $\hat{y}$ is binary(1 for relevant and 0 for irrelevant). S can be seen as the training set for this classification problem, 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 minimal size or even empty. Inspired by [8], our model computes $f^{\boldsymbol{\omega}}(d_q, S, d_i)$ as:

$$f^{\boldsymbol{\omega}}(d_q, S, d_i) = \begin{cases} \mathrm{c}^{\boldsymbol{\omega}}(d_q, d_i) & S = \varnothing \\ \mathrm{c}^{\boldsymbol{\omega}}(d_q, d_i) + \frac{1}{|S|}\sum_{(\hat{d},\hat{y})\in S} \mathrm{c}^{\boldsymbol{\omega}}(\hat{d}, d_i)\hat{y} & S \neq \varnothing \end{cases}, \tag{4}$$

where $c^{\boldsymbol{\omega}}(\cdot,\cdot)$ is our attention-based CNN for text similarity with parameters $\boldsymbol{\omega}$, 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 Fig. 2.

4.1.2 *Attention-Based CNN for Text Similarity*

Each article d_i is a sequence of l_i terms $[t_{i1}, t_{i2}, \ldots, t_{il_i}]$ (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, $\mathbf{M}^{(m,n)} \in \mathbb{R}^{l_m \times l_n}$, is defined as follows:

$$\mathbf{M}_{i,j}^{(m,n)} = \frac{\mathbf{w}_{mi}^T \cdot \mathbf{w}_{nj}}{\|\mathbf{w}_{mi}\| \cdot \|\mathbf{w}_{nj}\|}, \tag{5}$$

where $\mathbf{w}_{mi}$ and $\mathbf{w}_{nj}$ are the word embeddings of term t_{mi} and t_{nj}. Since the cosine similarity of word embeddings can capture the semantic similarity [54], $\mathbf{M}_{i,j}^{(m,n)}$ represents 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, thus cannot distinguish the matching signals of essential terms from those of structural, unimportant terms.

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, $\mathbf{A}^{(m,n)} \in \mathbb{R}^{l_m \times l_n}$ is defined as follows:

$$\mathbf{A}_{i,j}^{(m,n)} = r_{mi} \cdot r_{nj}, \tag{6}$$

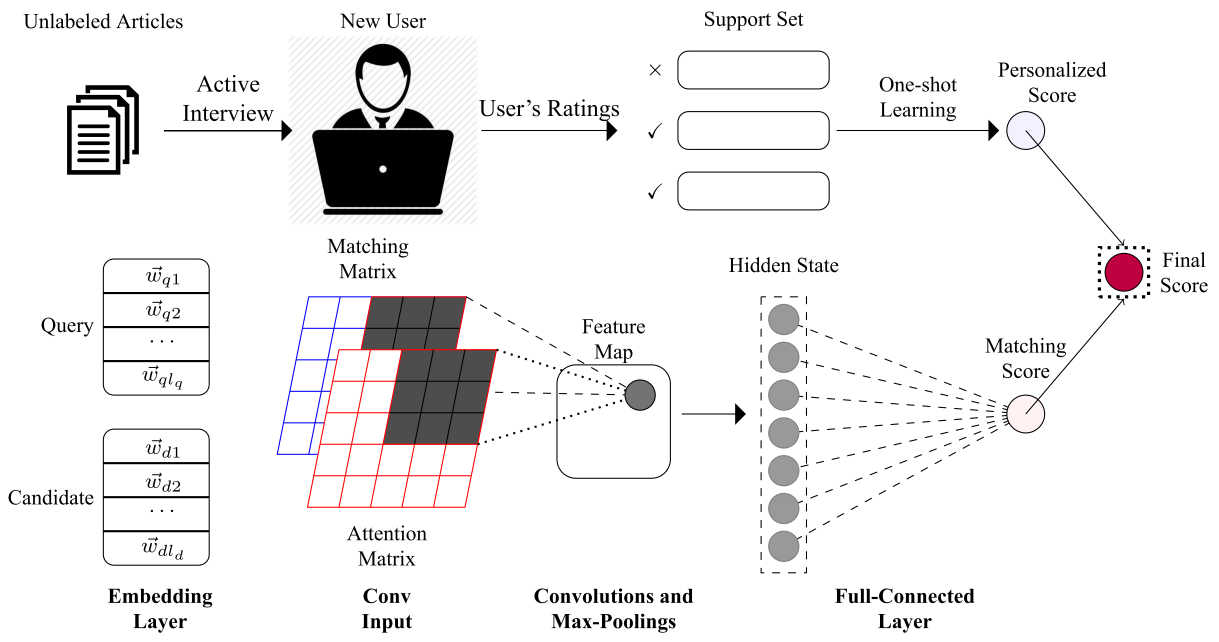


Fig. 2. The architecture of the overall framework. The articles are transformed into sequences of word embeddings through the embedding layer. The attention matrix and matching matrix are computed and sent to the CNN. The matching scores are combined with the support set gained from active interaction with the user to get the final scores.

where r_{mi} and r_{nj} are the weights of term t_{mi} and t_{nj}.

The matching matrix $\mathbf{M}^{(m,n)}$ and the attention matrix $\mathbf{A}^{(m,n)}$ are combined by element-wise multiplication:

$$\mathbf{Z}^{(m,n)} = \mathbf{M}^{(m,n)} \otimes \mathbf{A}^{(m,n)}, \tag{7}$$

$\mathbf{Z}^{(m,n)}$ is the input of a CNN that consists of several convolutional layers and max-pooling layers. Similar to CNNs in image recognition [58], 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. 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.

TABLE 1
Attention Mechanisms in CNN

Method	Description
Object Parts Selection	In *fine-grained classification*[55], image patches which contain parts of certain objects are selected through a supervised process to extract discriminative features.
Attention Matrix	In [56], an attention matrix is employed to give different attention weights to units in a feature map.
Configurable Convolution	For *visual question answering* task [57], configurable convolutional kernels are generated by transforming the question embeddings from the semantic space into the visual space, which implements the question-guided attention.

4.1.3 *Local Weight and Global Weight*

Traditional methods for text 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 IDF (inverse document frequency, the inverse of 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:

$$r_{ij} = \mu_{ij} \cdot \upsilon_{ij}, \tag{8}$$

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

Local Weight. The local weight measures the relevance of a term to the subject of the document. For example, in the following text [59]:

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 the term frequency (TF) in TF-IDF [25] or BM25 [26] ranking function:

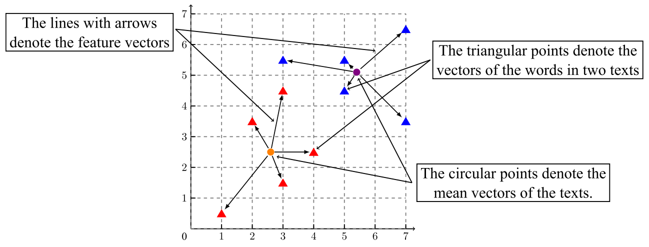

The lines with arrows denote the feature vectors
The triangular points denote the vectors of the words in two texts
The circular points denote the mean vectors of the texts.

Fig. 3. A two-dimensional example of the feature vectors for local weights.

$$\mathrm{BM25}(d,q)=\sum_{i=1}^{n}\mathrm{IDF}(q_i)\frac{f(q_i,d)(k_1+1)}{f(q_i,d)+k_1(1-b+b\frac{|d|}{l})},\tag{9}$$

where q_i is the ith term of the query, $f(q_i,d)$ is the term frequency of q_i in d and l 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.

Inspired by [60], 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 input vector $\mathbf{x}_{ij}$ for the local weight μ_{ij} is the difference between the word vector $\mathbf{w}_{ij}$ and the mean vector of d_i:

$$\mathbf{x}_{ij}=\mathbf{w}_{ij}-\overline{\mathbf{w}}_i,\tag{10}$$

where

$$\overline{\mathbf{w}}_i=\frac{1}{n_i}\sum_{k=1}^{n_i}\mathbf{w}_{ik},\tag{11}$$

Fig. 3 gives an illustration of the feature vector.

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

$$\mu_{ij}=\sigma(\mathbf{W}^{(L)}\cdot\mathbf{u}_{ij}^{(L)}+\mathbf{b}^{(L)})+\alpha,\tag{12}$$

where L is the number of hidden layers in the feed-forward network, $\mathbf{u}_{ij}^{(L)}$ is the output of the last hidden layer, 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 the maximum value to the 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. 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. However, 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:

$$\mathrm{IDF}(t)=\log\left(\frac{N}{n_t}\right),\tag{13}$$

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 did not 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:

$$\upsilon_{ij}=[\mathrm{IDF}(t_{ij})]^{\beta},\tag{14}$$

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

4.2 Bayesian Active Learning in Recommendation

4.2.1 Pairwise Loss for Bayesian Learning

We formulate the training set in the setting of *Pairwise Approach of Learning to Rank* [61]. It means that $\mathbb{D}_{train}=\{(d_q^{(i)}, S^{(i)}), d_+^{(i)}, d_-^{(i)}\}_{i=1}^{N}$, where article $d_+^{(i)}$ is ranked higher than article $d_-^{(i)}$ with respect to query article $d_q^{(i)}$ and support set $S^{(i)}$. We transform it into the constraint on the recommendation score as $y_+^{(i)}-y_-^{(i)}\geq 1$ and the log likelihood in the first term of (2) becomes:

$$\begin{aligned}&\log p(\mathbb{D}_{train}|f^{\boldsymbol{\omega}}(\mathbf{x}))\\&=\sum_{i=1}^{N}\log p(y_+^{(i)}-y_-^{(i)}\geq 1)|f^{\boldsymbol{\omega}}(\mathbf{x}_+^{(i)}),f^{\boldsymbol{\omega}}(\mathbf{x}_-^{(i)})\end{aligned}$$

Let f_+ and f_- denote $f^{\boldsymbol{\omega}}(\mathbf{x}_+)$ and $f^{\boldsymbol{\omega}}(\mathbf{x}_-)$

$$\begin{aligned}&p(y_+-y_-\geq 1|f_+,f_-)\\&=\int_{f_-}^{\infty}\int_{y_-+1}^{\infty}p(y_+|f_+)p(y_-|f_-)\mathrm{d}y_+\mathrm{d}y_-.\end{aligned}$$

The distribution of the score given model output, $p(y|f^{\boldsymbol{\omega}}(\mathbf{x}))$, partly decides the form of our objective function. We didn't use a Normal distribution like most cases, because the square difference of the Normal distribution can show an unstable behavior in the initial points of the Neural Network weight optimization. Instead, we define the distribution $p(y|f^{\boldsymbol{\omega}}(\mathbf{x}))$ as an Exponential Distribution:

$$p(y|f^{\boldsymbol{\omega}}(\mathbf{x}))=\begin{cases}\tau\exp(-\tau(y-f^{\boldsymbol{\omega}}(\mathbf{x}))) & y\geq f^{\boldsymbol{\omega}}(\mathbf{x})\\ 0 & y<f^{\boldsymbol{\omega}}(\mathbf{x})\end{cases},\tag{15}$$

where τ is the model precision.

With this distribution, we have:

$$\begin{aligned}&p(y_+-y_-\geq 1|f_+,f_-)\\&=\int_{f_-}^{\infty}\int_{y_-+1}^{\infty}\tau^2\exp(-\tau(y_+-f_++y_--f_-))\mathrm{d}y_+\mathrm{d}y_-.\end{aligned}$$

When $f_+ - f_- \leq 1$, it is equal to

$$\begin{aligned}&\int_{f_-}^{\infty} \tau\exp(-\tau(2y_- - f_+ - f_- + 1))\mathrm{d}y_- \\ =&\frac{1}{2}\exp(-\tau(f_- - f_+ + 1)).\end{aligned}$$

When $f_+ - f_- > 1$, it is equal to

$$\begin{aligned}&\int_{f_-}^{f_+-1}\int_{f_+}^{\infty} \tau^2\exp(-\tau(y_+ - f_+ + y_- - f_-))\mathrm{d}y_+\mathrm{d}y_- \\ &+\int_{f_+-1}^{\infty}\int_{y_-+1}^{\infty} \tau^2\exp(-\tau(y_+ - f_+ + y_- - f_-))\mathrm{d}y_+\mathrm{d}y_- \\ =&1-\exp(-\tau(f_+ - f_- - 1)) + \frac{1}{2}\exp(-\tau(f_+ - f_- - 1)) \\ =&1-\frac{1}{2}\exp(-\tau(f_+ - f_- - 1)).\end{aligned}$$

Therefore we have

$$\log\left(\mathbb{D}_{train}|f^{\boldsymbol{\omega}}(\mathbf{x})\right) = \sum_{i=1}^{N} E^{\boldsymbol{\omega}}[(d_q^{(i)}, S^{(i)}), d_+^{(i)}, d_-^{(i)}],$$

where

$$\begin{aligned}&E^{\boldsymbol{\omega}}[(d_q, S), d_+, d_-] \\ =&\begin{cases}\tau(1 - f_+ + f_-) & f_+ - f_- \leq 1 \\ \log\left(1 - \frac{1}{2}\exp(-\tau(f_+ - f_- - 1))\right) & f_+ - f_- > 1\end{cases}.\end{aligned} \tag{16}$$

When $f_+ - f_- \leq 1$, this function is the same as the hinge loss function except for the coefficient τ. However, when $f_+ - f_- > 1$, the loss will not directly drop to zero, but gradually decrease to zero as $f_+ - f_-$ increases. Therefore, it can be considered as the smoothed version of hinge loss.

As for the second term in (2), it's proved in [42] that it can be approximated by L2 regularization term $\sum_{\mathbf{W}_i\in\boldsymbol{\omega}} \lambda_i||\mathbf{W}_i||^2$, as long as the weight decay λ_i satisfies:

$$\lambda_i = \frac{1-p_i}{2\sigma_i^2}, \tag{17}$$

where $p(\mathbf{W}_i) \sim \mathcal{N}(0, \sigma_i^2\mathbf{I})$.

Above all, maximizing (2) can be approximated as minimizing the following loss function

$$\mathcal{L} = \int -q(\boldsymbol{\omega})\sum_{i=1}^{N} E^{\boldsymbol{\omega}}(\mathbb{D}_{train}^{(i)})\mathrm{d}\boldsymbol{\omega} + \sum_{\mathbf{W}_i\in\boldsymbol{\omega}} \lambda_i||\mathbf{W}_i||^2. \tag{18}$$

The final problem is, to minimize the above loss function, we have to integrate over the parameter space. The integration can be approximated by Monte-Carlo Sampling from $q(\boldsymbol{\omega})$. For a neural network with dropout, Monte-Carlo Sampling is equivalent to forward pass with dropout. Therefore, minimizing the objective in (2) is equivalent to optimizing the following loss function in neural networks:

$$\mathcal{L} = -\sum_{i=1}^{N} E^{\boldsymbol{\omega}}(\mathbb{D}_{train}^{(i)}) + \sum_{\mathbf{W}_i\in\boldsymbol{\omega}} \lambda_i||\mathbf{W}_i||^2. \tag{19}$$

After training, forward pass with dropout through the network is equivalent to sampling from the optimal distribution $q^*(\boldsymbol{\omega})$. This leads to a Monte-Carlo integration to compute the predicted distribution:

$$\begin{aligned}p(y|\mathbf{x}, \mathbb{D}_{train}) &= \int p(y|f^{\boldsymbol{\omega}}(\mathbf{x}))p(\boldsymbol{\omega}|\mathbb{D}_{train})\mathrm{d}\boldsymbol{\omega} \\ &\approx \int p(y|f^{\boldsymbol{\omega}}(\mathbf{x}))q^*(\boldsymbol{\omega})\mathrm{d}\boldsymbol{\omega} \\ &\approx \frac{1}{T}\sum_{t=1}^{T} p(y|f^{\hat{\boldsymbol{\omega}}_t}(\mathbf{x})),\end{aligned}$$

where $\hat{\boldsymbol{\omega}}_t \sim q^*(\boldsymbol{\omega})$, which is the model parameter sampled from optimal dropout distribution.

The optimization is done through standard backpropagation [62] and stochastic gradient descent method with minibatches. For regularization, we use dropout in the output of every hidden layer and early stopping strategy [63] to avoid over-fitting.

4.2.2 Expected Loss Optimization

Algorithm 1. Active Personalized Article Recommendation Algorithm

```
Input: A new user $u$, the query document $d_q$, an unlabeled set
  $\mathcal{U}$, the query budget $b$.
Output: The support set $S$ for $u$
  $S \leftarrow \emptyset$
  for i=1to N do {N=size of unlabeled set}
      $a_i \leftarrow \sum_{j=1}^{N} c(d_i, d_j)$
  end for
  for n=1to b do
    for $i = 1$ to $T$ do {T=number of MC sampling}
      for $j = 1$ to $N$ do
          Take forward pass through the network with dropout
          and get the output $y_j^i = f(d_q, S, d_i)$
      end for
    end for
    for $j = 1$ to $N$ do
      EL($j$) $\leftarrow 0$
      $g_j \leftarrow \frac{\sum_{k=1}^{N} y_j^k}{N}$
      for $i = 1$ to $T$ do
        $d_j^i \leftarrow$ BDCG($\{y_k^i\}_{k=1}^{N}$))
        for $k = 1$ to $N, k \neq j$ do
          $g_k \leftarrow y_k^i$
        end for
        EL($j$) $\leftarrow$ EL($j$) $+ d_j^i -$ BDCG($\{g_k\}_{k=1}^{N}$)
      end for
    end for
    $i \leftarrow \text{argmax}_{(d_j, u(d_j) \notin S)}\, a_j \cdot$ EL($j$)
    $S \leftarrow S \cup \{(d_i, u(d_i))\}$
  end for
```

The active learning metric we choose is Expected Loss Optimization [17]. The basic idea is to choose the instance that *maximizes the expected loss of the current best action*. For the ranking problem, the action for an instance refers to deciding its ranking in the list. Therefore the expected loss for article d_i given d_q and S is

$$\begin{aligned}&\mathrm{EL}(d_i|d_q,S)\\&=\int_{\mathbf{Y}^i}\min_{\pi}\int_{y_i} l(\pi,\mathbf{Y})P(\mathbf{Y}|(d_q,S),\mathbb{D}_{train})\mathrm{d}y_i\mathrm{d}\mathbf{Y}^i,\end{aligned}\tag{20}$$

where $l(\cdot,\cdot)$ is the loss function, $\mathbf{Y}$ is the vector of the recommendation scores of all the documents and $\mathbf{Y}^i$ is $\mathbf{Y}$ after removing y_i.

For ranking problem, we define the loss function as the difference between the DCG for current ranking and the ranking with the largest DCG:

$$l(\pi,\mathbf{Y})=\max_{\pi'}\mathrm{DCG}(\pi',\mathbf{Y})-\mathrm{DCG}(\pi,\mathbf{Y}),\tag{21}$$

where $\mathrm{DCG}(\pi,\mathbf{Y})=\sum_i\frac{y_i}{\log_2(1+\pi(i))}$.

Combining (20) and (21), we have:

$$\begin{aligned}&\mathrm{EL}(d_i|d_q,S)\\&=\int_{\mathbf{Y}^i}[\int_{y_i}\max_{\pi}\mathrm{DCG}(\pi,\mathbf{Y})P(\mathbf{Y}|(d_q,S),\mathbb{D}_{train})\mathrm{d}y_i\\&-\max_{\pi}\int_{y_i}\mathrm{DCG}(\pi,\mathbf{Y})P(\mathbf{Y}|(d_q,S),\mathbb{D}_{train})\mathrm{d}y_i]\mathrm{d}\mathbf{Y}^i.\end{aligned}\tag{22}$$

An important property of expected loss for ranking is that it considers not only the uncertainty of the article's predicted score, but also its current ranking among unlabeled articles. With the same predicted uncertainty, the article with a higher ranking has a higher expected loss. In this way, ELO surpasses traditional uncertainty sampling, which only considers the prediction uncertainty.

However, ELO can still be troubled by outliers, the instances that are located in the sparse area of the input space. We further propose to combine the density with ELO:

$$\arg\max_{d_i}\mathrm{EL}(d_i|d_q,S)\times\frac{1}{|\mathcal{U}|}\sum_{d_j\in\mathcal{U}}c(d_i,d_j),\tag{23}$$

where $c(d_i,d_j)$ is the similarity between d_i and d_j predicted by our CNN model.

The latter part in (23) is the average similarity between d_i and articles in $\mathcal{U}$. (23) aims to find the instance that is representative of most unlabeled articles while maximizing the expected loss at the same time. The completed algorithm for active learning is described in Algorithm 1.

5 Dataset and Experiment Setting

To evaluate the proposed model, we conduct experiments on article recommendation problem in non-personalized setting, personalized setting, and active setting, based on datasets of different sources and scales, in comparison with baselines in both article recommendation and active learning. In this section, we will introduce the dataset and experiment setting.

5.1 Dataset

We evaluate the performance of the proposed model on article recommendation two small, manually labeled datasets and a large-scale dataset based on user clicks.

The first dataset is based on papers from AMiner [1] and consists of 188 query papers with ten 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[3] 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 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.

The third dataset is Related-Article Recommendation Dataset (RARD) [64] 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. A recommender-as-a-service provider Mr. DLib generates the displayed documents, 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.

To conduct experiments on Active One-shot Article Recommendation Problem, we use the Citation Network Dataset in AMiner [1]. The citation data is extracted from DBLP, ACM, MAG (Microsoft Academic Graph), and other sources. The version we used contains 3,272,991 papers and 8,466,859 citations. The citations of a paper are randomly divided into two equal parts. The first part is used as the support set(or unlabeled set in Active Learning setting) and the second part is used as the positive recommendations for training and evaluation. To ensure the quality, we randomly select the negative recommendations from the neighborhood of the paper in the citation network. Abstracts of papers are used as texts and texts longer than 96 terms are truncated.

5.2 Baselines for Article Recommendation

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 [65]*: 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 [66]*: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 [13]*:The interactions between different positional sentence representations generated by a Bi-LSTM form a similarity matrix to generate the matching score.
- *MatchPyramid [12]*: A CNN is built on the standard matching matrix to get the matching score.
- *DRMM [28]*:The matching between the terms in the query and the document is expressed as a histogram,

3. http://patft.uspto.gov/

TABLE 2
Results of Relevance Ranking (%)

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**

NG stands for NDCG.

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 [14]*:An interaction-based model and a representation-based model are combined to get the matching score of two articles.

For the fairness of comparison, all models don't involve user feedback, which will be discussed in Section 6.

5.3 Baselines for Active Learning

- *Random*: The active documents are selected randomly from the unlabeled set. Any method that can't beat this baseline doesn't make sense.
- *Entropy*: The entropy is a standard uncertainty measure for classification gained from information theory. We define the recommendation problem as binary classification to predict the recommendation score is 0 or 1 and $p_i = p(y_i = 1)$, then

$$H[y_i] = -[p_i \log p_i + (1-p_i)\log(1-p_i)].$$

- *Relevance*: For this method we always choose the document that has the largest recommendation score, which means it's the most relevant to the user's known preference.
- *Expected Error Reduction [38]*: The active selection strategy is to maximize the expected reduction of the loss function after retraining the model on the new training set. We use the cross entropy as the loss function.
- *BALD [40]*: The active selection strategy is to maximize the expected reduction of parameter entropy after retraining the model on the new training set. By removing the part unrelated to $\vec{x}$ and rearrange the order of integral, the objective can become entropies in y space:

$$\underset{\mathbf{x}}{argmax}\, \mathrm{H}(y|\mathbf{x}, \mathbb{D}_{train}) - \mathbb{E}_{\boldsymbol{\omega}\sim p(\boldsymbol{\omega}|D_{train})}[H[y|f^{\boldsymbol{\omega}}(\mathbf{x})]].$$

The following are several variations of our proposed model:

- *POLAR++Variance*: The variance is a common uncertainty measure for regression. We use $\mathbb{E}[\cdot]$ to denote $\mathbb{E}_{\boldsymbol{\omega}\sim p(\boldsymbol{\omega}|\mathbb{D}_{train}), y\sim p(y|f^{\boldsymbol{\omega}}(\mathbf{x}))}[\cdot]$

$$\begin{aligned}&\mathrm{Var}(y|\mathbf{x}, \mathbb{D}_{train})\\ =&\mathbb{E}[(y-\mathbb{E}[y|\boldsymbol{\omega}])^2] + \mathbb{E}[(\mathbb{E}[y|\boldsymbol{\omega}] - \mathbb{E}[y])^2].\end{aligned}$$

The first term is a constant $(\frac{1}{\tau})^2$ dependent on the model precision. The second term can be approximated by the variance of predicted y in Monte-Carlo Sampling.

- *POLAR++ELO*The active selection strategy is to maximize the Expected Loss for Ranking without density according to (22).
- *POLAR++DWELO*DWELO (Density-Weighted Expected Loss Optimization) is the complete active leraning algorithm which combines ELO and density, as described in Algorithm 1.

5.4 Parameter Setting

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 [54].

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 6.

We set T in Algorithm 1, the number of MC sampling, as 32, to balance the precision and time complexity.

6 Result and Analysis

6.1 Non-Persoalized Setting

In the non-personalized setting, only the query article is given while the support set is kept empty. Table 2 shows the ranking accuracy of different methods in terms of NDCG.

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 percent on NDCG@3 and 3.3-20.3 percent on NDCG@5. The average improvements of NDCG on each dataset are respectively 3.8, 8.1 and 6.4 percent.

Among the traditional ranking models, TF-IDF is the most competitive one, even outperforming the best neural baselines by 5.5 percent in some cases. 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

TABLE 3
Performance for the Model With One-Shot Learning and Without

	AMiner		Patent		RARD	
Method	NG@1	NG@3	NG@1	NG@3	NG@1	NG@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**

NG stands for NDCG.

person to person. Only taking the exact matching signals into account, TF-IDF might be unsuitable for such a 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 Match Pyramid, 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.

6.2 Personalized Setting

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 the support set is only 1 for 45 percent queries and 2 for 47 percent. In the patent dataset, the sizes of 75 percent support sets 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 percent on NDCG@1 and 5.7 percent on NDCG@3.

6.3 Active Learning Setting

In this section, we show that the proposed active learning method can effectively select informative articles to improve recommendation performance in the active learning setting. For each recommendation episode, the support set is empty at the beginning. At each step, we select an unlabeled article according to the evaluated method and add the article and corresponding label to the support set. Following the traditional evaluation method for active learning, we show the NDCG of different strategies as the function of rated articles. Fig. 4 shows the NDCG comparison results on the Citation Dataset.

We can see that POLAR++DWELO can outperform all the baselines w.r.t. NDCG@n metrics. As n increases, the margin between our method and baselines continuously increases. This is also reasonable because when n is small, even the worst strategy can find a few articles that the user must be interested in. A wise strategy aims to find the complete preferences of the user.

All the baselines can achieve better performance than the Random strategy, which proves their utility in active learning. Among all the non-random baselines, the Relevance strategy performs worst, because the Relevance strategy always chooses the article that is most relevant to the user's known preferences, but often less informative.

The Expected Error Reduction method can perform best at first but soon fails. As the number of rated articles increases, its performance even falls behind Relevance strategy. This is because it chooses the article which can reinforce the existing belief over the unlabeled articles. Therefore it can be restricted to the user's partial interest.

For uncertainty based method, BALD performs slightly better than Entropy. This is quite surprising because BALD is derived from minimizing the entropy of parameter posterior, which does not make sense in our one-shot learning setting.

6.4 More Analysis

6.4.1 Comparison of Different Variations of POLAR++

For simplicity, we only show the result of the complete algorithm, POLAR++DWELO, along with those of baselines, in Fig. 4. The NDCG comparison results among different variations of our proposed method are shown in Fig. 5.

Both without the help of density information, POLAR++ELO can significantly outperform POLAR++Variance. This confirms our idea in Section 4.2.2: Expected Loss Optimization is a better active learning strategy for the ranking problem than Uncertainty Sampling, because it considers not only the prediction uncertainty but also the relative ranking in the list. The density-weighted version of ELO, POLAR++DWELO, can further perform slightly better than POLAR++ELO. This proves that with the help of density information, the algorithm can choose the informative articles, rather than outliers, the articles that are not representative in document space.

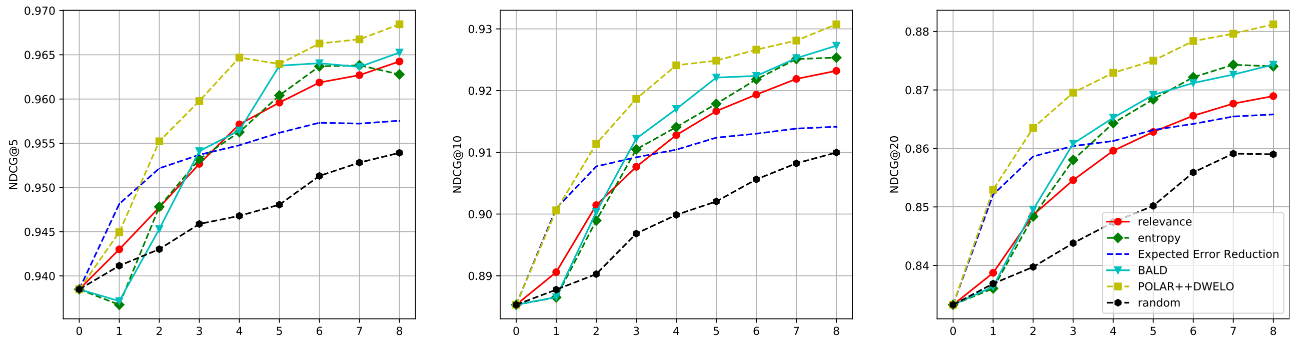


Fig. 4. NDCG@5,10, and 20 as the functions of queried articles for different active learning methods.

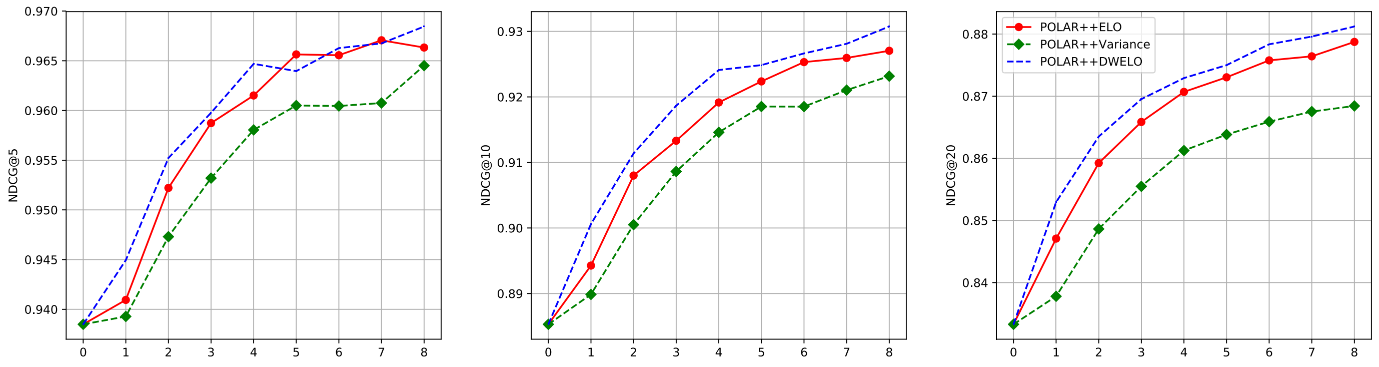


Fig. 5. NDCG@5, 10, and 20 as the functions of queried articles for different variations of POLAR++.

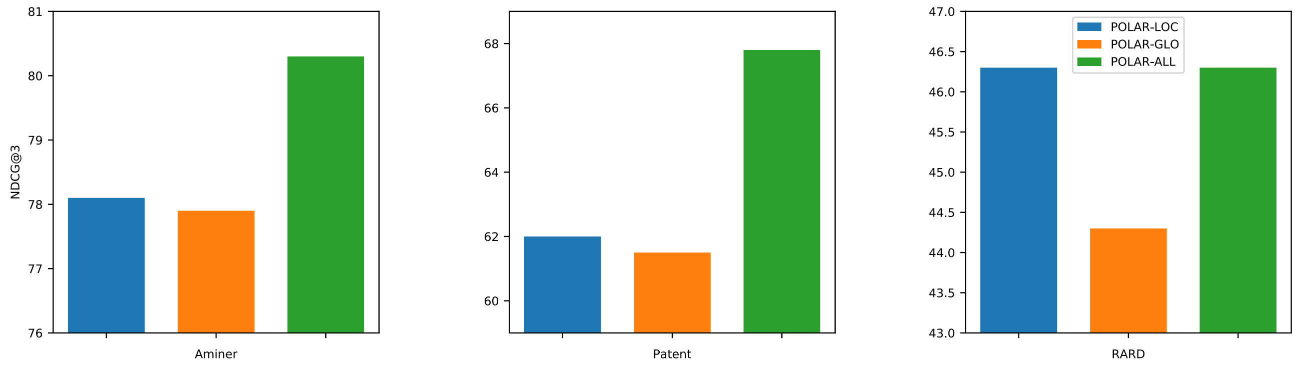


Fig. 6. The performance of different attention matrices.

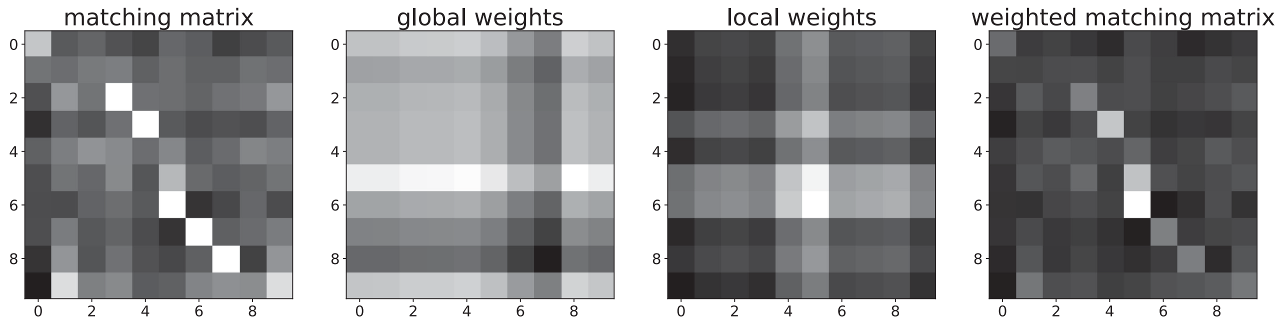


Fig. 7. The visualization result of four matrices used in the matching of a pair of texts. The brighter the pixel is, the larger value it has. The text pair is as follows(the words in brackets are removed stopwords): T1:novel robust stability criteria (for) stochastic hopfield neural networks (with) time delays; T2:new delay dependent stability criteria (for) neural networks (with) time varying delay.

6.4.2 *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 Fig. 6.

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 global weights. The complete model, POLAR-ALL, which combines the two weights, performs significantly better than either of them. This confirms that the local and global weights are complementary to each other. However, we also see that in RARD, the performances of POLAR-LOC and POLAR-All are quite close, which is against the result on other datasets. We guess the reason might be that RARD dataset is in German and contains less unlabeled texts. This can lead to the inaccurate global weights based on IDF values and as a result, global weight matrix cannot help much in POLAR-ALL.

To have a better understanding of how local and global weights work, we show the pixel images of four matrices in Fig. 7. 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

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

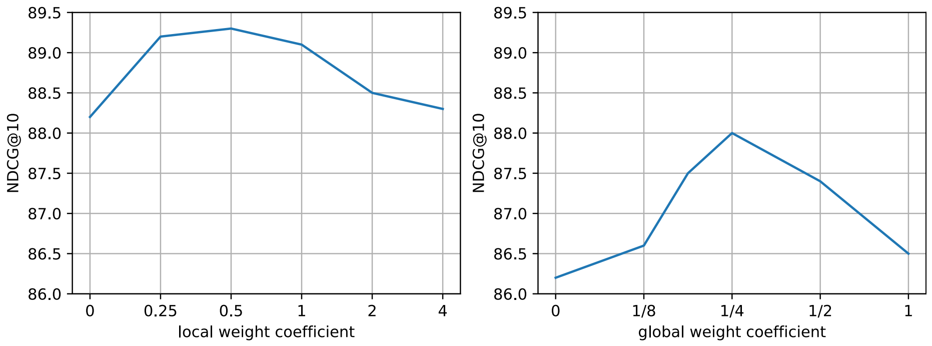


Fig. 8. Performance comparison for POLAR-LOC with different α and POLAR-GLO with different β on the AMiner dataset.

terms in the corpus with low weights, while the local weights function by highlighting key terms in specific articles.

6.4.3 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 Fig. 8 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 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.

7 Conclusion and Future Work

In this paper, we study the problem of actively learning users' preference in article recommendation. We define the Active One-shot Article Recommendation Problem, which is extended from our previous definition of One-shot Personalized Article Recommendation. We propose a novel framework POLAR++ in which an active learning algorithm based Bayesian NN is applied to deal with the user cold-start issue. An attention-based CNN model for text similarity is combined with the framework of one-shot learning to deal with sparse user feedback. 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 recommendations of similar papers.

The limit of our model is that it combines the information of different support articles at a high level. Our future work might consider combining the information of different support articles at lower levels, such as iterating over the support set with an LSTM or CNN. Moreover, we would like to combine our model with Reinforcement Learning to train a deeper and more powerful model in the online environment.

Acknowledgments

The work is supported by the National Key R&D Program of China (2018YFB1402600), NSFC for Distinguished Young Scholar (61825602), and Tsinghua University Initiative Scientific Research Program (20181080300).

References

[1] J. Tang, J. Zhang, L. Yao, J. Li, L. Zhang, and Z. Su, "ArnetMiner: Extraction and mining of academic social networks," in *Proc. 14th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2008, pp. 990–998.
[2] F. Zhang *et al.*, "OAG: Toward linking large-scale heterogeneous entity graphs," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2019, pp. 2585–2595.
[3] Y. Qi, Q. Wu, H. Wang, J. Tang, and M. Sun, "Bandit learning with implicit feedback," in *Proc. 32nd Int. Conf. Neural Inf. Process. Syst.*, 2018, pp. 7287–7297.
[4] M. J. Pazzani and D. Billsus, "Content-based recommendation systems," in *The Adaptive Web, Methods and Strategies of Web Personalization*, Berlin, Germany: Springer, 2007, pp. 325–341.
[5] J. S. Breese, D. Heckerman, and C. Kadie, "Empirical analysis of predictive algorithms for collaborative filtering," in *Proc. 14th Conf. Uncertainty Artif. Intell.*, 1998, pp. 43–52.
[6] G. Koch, R. Zemel, and R. Salakhutdinov, "Siamese neural networks for one-shot image recognition," in *Proc. ICML Deep Learn. Workshop*, 2015, vol. 2.
[7] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. P. Lillicrap, "Meta-learning with memory-augmented neural networks," in *Proc. 33rd Int. Conf. Int. Conf. Mach. Learn.*, 2016, pp. 1842–1850.
[8] O. Vinyals, C. Blundell, T. Lillicrap, K. Kavukcuoglu, and D. Wierstra, "Matching networks for one shot learning," in *Proc. 30th Int. Conf. Neural Inf. Process. Syst.*, 2016, pp. 3630–3638.
[9] C. Wang and D. M. Blei, "Collaborative topic modeling for recommending scientific articles," in *Proc. 17th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2011, pp. 448–456.
[10] J. Beel, B. Gipp, S. Langer, and C. Breitinger, "Research paper recommender systems: A literature survey," *Int. J. Digital Libraries*, vol. 17, no. 4, pp. 305–338, 2016.
[11] 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 *Proc. 22nd ACM Int. Conf. Inf. Knowl. Manage.*, 2013, pp. 2333–2338.
[12] L. Pang, Y. Lan, J. Guo, J. Xu, S. Wan, and X. Cheng, "Text matching as image recognition," in *Proc. 30th AAAI Conf. Artif. Intell.*, 2016, pp. 2793–2799.
[13] S. Wan, Y. Lan, J. Guo, J. Xu, L. Pang, and X. Cheng, "A deep architecture for semantic matching with multiple positional sentence representations," in *Proc. 30th AAAI Conf. Artif. Intell.*, 2016, pp. 2835–2841.
[14] B. Mitra, F. Diaz, and N. Craswell, "Learning to match using local and distributed representations of text for web search," in *Proc. 26th Int. Conf. World Wide Web*, 2017, pp. 1291–1299.
[15] C. Xiong, Z. Dai, J. Callan, Z. Liu, and R. Power, "End-to-end neural ad-hoc ranking with kernel pooling," in *Proc. Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2017, pp. 55–64.
[16] Z. Du, J. Tang, and Y. Ding, "POLAR: Attention-based CNN for one-shot personalized article recommendation," in *Proc. Joint Eur. Conf. Mach. Learn. Knowl. Discovery Databases*, 2018, pp. 675–690.
[17] B. Long, J. Bian, O. Chapelle, Y. Zhang, Y. Inagaki, and Y. Chang, "Active learning for ranking through expected loss optimization," *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 5, pp. 1180–1191, May 2015.

[18] K. D. Bollacker, S. Lawrence, and C. L. Giles, “CiteSeer: An autonomous web agent for automatic retrieval and identification of interesting publications,” in *Proc. Int. Conf. Auton. Agents*, 1998, pp. 116–123.

[19] Q. He, J. Pei, D. Kifer, P. Mitra, and C. L. Giles, “Context-aware citation recommendation,” in *Proc. 19th Int. Conf. World Wide Web*, 2010, pp. 421–430.

[20] F. Ferrara, N. Pudota, and C. Tasso, “A keyphrase-based paper recommender system,” in *Proc. 7th Italian Res. Conf. Digital Libraries Archives*, 2011, pp. 14–25.

[21] Y. Jiang, A. Jia, Y. Feng, and D. Zhao, “Recommending academic papers via users’ reading purposes,” in *Proc. 6th ACM Conf. Recommender Syst.*, 2012, pp. 241–244.

[22] D. M. Pennock, E. Horvitz, S. Lawrence, and C. L. Giles, “Collaborative filtering by personality diagnosis: A hybrid memory and model-based approach,” in *Proc. 16th Conf. Uncertainty Artif. Intell.*, 2000, pp. 473–480.

[23] M. Gori and A. Pucci, “Research paper recommender systems: A random-walk based approach,” in *Proc. Int. Conf. Web Intell.*, 2006, pp. 778–781.

[24] N. Lao and W. W. Cohen, “Relational retrieval using a combination of path-constrained random walks,” *Mach. Learn.*, vol. 81, no. 1, pp. 53–67, 2010.

[25] G. Salton, E. A. Fox, and H. Wu, “Extended boolean information retrieval,” *Commun. ACM*, vol. 26, no. 11, pp. 1022–1036, 1983.

[26] S. E. Robertson, S. Walker, S. Jones, M. Hancock-Beaulieu, and M. Gatford, “Okapi at TREC-3,” in *Proc. Text REtrieval Conf.*, 1994, pp. 109–126.

[27] H. Palangi *et al.*, “Deep sentence embedding using long short-term memory networks: Analysis and application to information retrieval,” *IEEE/ACM Trans. Audio Speech Language*, vol. 24, no. 4, pp. 694–707, Apr. 2016.

[28] J. Guo, Y. Fan, Q. Ai, and W. B. Croft, “A deep relevance matching model for ad-hoc retrieval,” in *Proc. Int. Conf. Inf. Knowl. Manage.*, 2016, pp. 55–64.

[29] F. Li, R. Fergus, and P. Perona, “One-shot learning of object categories,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 4, pp. 594–611, Apr. 2006.

[30] S. Ravi and H. Larochelle, “Optimization as a model for few-shot learning,” in *Proc. Int. Conf. Learn. Representations*, 2017, vol. 2, Art. no. 6.

[31] V. Federov, *Theory of Optimal Experiments*. Cambridge, MA, USA: Academic Press, 1972.

[32] D. Angluin, “Queries and concept learning,” *Mach. Learn.*, vol. 2, no. 4, pp. 319–342, 1987.

[33] L. E. Atlas, D. A. Cohn, and R. E. Ladner, “Training connectionist networks with queries and selective sampling,” in *Proc. 2nd Int. Conf. Neural Inf. Process. Syst.*, 1989, pp. 566–573.

[34] D. D. Lewis and W. A. Gale, “A sequential algorithm for training text classifiers,” in *Proc. 17th Annu. Int. ACM SIGIR Conf. Res. Development Information Retrieval*, 1994, pp. 3–12.

[35] C. E. Shannon, “A mathematical theory of communication,” *Bell System Tech. J.*, vol. 27, pp. 379–423, 1948.

[36] D. A. Cohn, Z. Ghahramani, and M. I. Jordan, “Active learning with statistical models,” in *Proc. Conf. Neural Inf. Process. Syst.*, 1994, pp. 705–712.

[37] Y. Freund, H. S. Seung, E. Shamir, and N. Tishby, “Selective sampling using the query by committee algorithm,” *Mach. Learn.*, vol. 28, no. 2–3, pp. 133–168, 1997.

[38] N. Roy and A. McCallum, “Toward optimal active learning through sampling estimation of error reduction,” in *Proc. Int. Conf. Mach. Learn.*, 2001, pp. 441–448.

[39] Y. Guo and R. Greiner, “Optimistic active-learning using mutual information,” in *Proc. 20th Int. Joint Conf. Artif. Intell.*, 2007, pp. 823–829.

[40] N. Houlsby, F. Huszar, Z. Ghahramani, and M. Lengyel, “Bayesian active learning for classification and preference learning,” 2011, *arXiv preprint arXiv:1112.5745*.

[41] Y. Gal, R. Islam, and Z. Ghahramani, “Deep bayesian active learning with image data,” in *Proc. 34th Int. Conf. Mach. Learn.*, 2017, pp. 1183–1192.

[42] Y. Gal and Z. Ghahramani, “Dropout as a bayesian approximation: Representing model uncertainty in deep learning,” in *Proc. 33rd Int. Conf. Int. Conf. Mach. Learn.*, 2016, pp. 1050–1059.

[43] R. Jin and L. Si, “A bayesian approach toward active learning for collaborative filtering,” in *Proc. 20th Conf. Uncertainty Artif. Intell.*, 2004, pp. 278–285.

[44] N. Golbandi, Y. Koren, and R. Lempel, “Adaptive bootstrapping of recommender systems using decision trees,” in *Proc. 4th ACM Int. Conf. Web Search Data Mining*, 2011, pp. 595–604.

[45] A. Harpale and Y. Yang, “Personalized active learning for collaborative filtering,” in *Proc. 31st Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2008, pp. 91–98.

[46] M. Nadimi-Shahraki and M. Bahadorpour, “Cold-start problem in collaborative recommender systems: Efficient methods based on ask-to-rate technique,” *J. Comput. Inf. Technol.*, vol. 22, no. 2, pp. 105–113, 2014.

[47] M. Elahi, F. Ricci, and N. Rubens, “A survey of active learning in collaborative filtering recommender systems,” *Comput. Sci. Rev.*, vol. 20, pp. 29–50, 2016.

[48] N. Rubens and M. Sugiyama, “Influence-based collaborative active learning,” in *Proc. ACM Conf. Recommender Syst.*, 2007, pp. 145–148.

[49] N. Rubens, R. Tomioka, and M. Sugiyama, “Output divergence criterion for active learning in collaborative settings,” *Inf. Media Technol.*, vol. 5, no. 1, pp. 119–128, 2010.

[50] T. Hofmann, “Collaborative filtering via gaussian probabilistic latent semantic analysis,” in *Proc. 26th Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2003, pp. 259–266.

[51] Y. Zhu *et al.*, “Addressing the item cold-start problem by attribute-driven active learning,” *IEEE Trans. Knowl. Data Eng.*, 2019.

[52] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe, “Variational inference: A review for statisticians,” *J. Amer. Statistical Assoc.*, vol. 112, no. 518, pp. 859–877, 2017.

[53] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: A simple way to prevent neural networks from overfitting,” *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.

[54] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, “Distributed representations of words and phrases and their compositionality,” in *Proc. 26th Int. Conf. Neural Inf. Process. Syst.*, 2013, pp. 3111–3119.

[55] T. Xiao, Y. Xu, K. Yang, J. Zhang, Y. Peng, and Z. Zhang, “The application of two-level attention models in deep convolutional neural network for fine-grained image classification,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 842–850.

[56] W. Yin, H. Schütze, B. Xiang, and B. Zhou, “ABCNN: Attention-based convolutional neural network for modeling sentence pairs,” *Trans. Assoc. Comput. Linguistics*, vol. 4, pp. 259–272, 2016.

[57] K. Chen, J. Wang, L.-C. Chen, H. Gao, W. Xu, and R. Nevatia, “ABC-CNN: An attention based convolutional neural network for visual question answering,” 2015, *arXiv preprint arXiv:1511.05960*.

[58] M. D. Zeiler and R. Fergus, “Visualizing and understanding convolutional networks,” in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 818–833.

[59] A. Das, “Audio visual person authentication by multiple nearest neighbor classifiers,” in *Proc. Int. Conf. Biometrics*, 2007, pp. 1114–1123.

[60] G. Zheng and J. Callan, “Learning to reweight terms with distributed representations,” in *Proc. 38th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2015, pp. 575–584.

[61] C. J. C. Burges *et al.*, “Learning to rank using gradient descent,” in *Proc. 22nd Int. Conf. Mach. Learn.*, 2005, pp. 89–96.

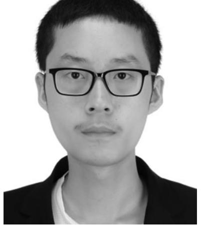
[62] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning representations by back-propagating errors,” *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[63] R. Caruana, S. Lawrence, and L. Giles, “Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping,” in *Proc. Conf. Neural Inf. Process. Syst.*, 2000, pp. 381–387.

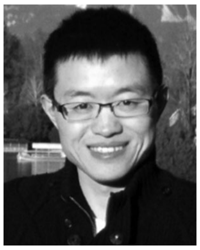
[64] J. Beel, Z. Carevic, J. Schaible, and G. Neusch, “RARD: The related-article recommendation dataset,” 2017, *arXiv preprint arXiv:1706.03428*.

[65] Q. V. Le and T. Mikolov, “Distributed representations of sentences and documents,” in *Proc. 31st Int. Conf. Mach. Learn.*, 2014, pp. 1188–1196.

[66] M. J. Kusner, Y. Sun, N. I. Kolkin, and K. Q. Weinberger, “From word embeddings to document distances,” in *Proc. 32nd Int. Conf. Mach. Learn.*, 2015, pp. 957–966.



Zhengxiao Du is working toward the senior undergraduate degree with the Department of Computer Science and Technology, Tsinghua University, Beijing, China. His main research interests include deep learning, reinforcement learning and their applications in recommender systems. He has published papers in PKDD/ECML.



Jie Tang is a full professor with the Department of Computer Science and Technology, Tsinghua University, Beijing, China. His main research interests include data mining algorithms and social network theories. He has been a visiting scholar with Cornell University, Chinese University of Hong Kong, Hong Kong University of Science and Technology, and Leuven University. He has published more than 200 research papers in major international journals and conferences including: KDD, IJCAI, AAAI, ICML, WWW, SIGIR, SIGMOD, ACL, the *Machine Learning Journal*, *ACM Transactions on Knowledge Discovery from Data*, and *IEEE Transactions on Knowledge and Data Engineering*.

Yuhui Ding is currently working toward the senior undergraduate degree with the Department of Computer Science and Technology, Tsinghua University, Beijing, China. His main research interests include recommender systems and text mining.