2014 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2014)

Personalized Paper Recommendation in Online Social Scholar System

Huan Xue, Jiafeng Guo, Yanyan Lan, Lei Cao
Institute of Computing Technology, CAS
Beijing, China
xuehuan,leicao@software.ict.ac.cn guojiafeng,lanyanyan@ict.ac.cn

***Abstract*—This paper presents a practical paper recommender system, which aims to provide personalized research paper recommendations to users within an online social scholar system. As an online recommender system, there are three basic problems we need to tackle: 1) How to formalize and solve the recommendation problem; 2) How to achieve real time recommendation; and 3) How to interact with users. In our work, we take the personalized paper recommendation as a ranking problem with respect to users' research interests, and employ a supervised learning to rank approach to solve the problem. However, most previous learning to rank methods rely on manually labeled training data which are both expensive and limited in size. We propose automatical training data construction by mining the existing large scale academic network, and extract various heterogeneous features for learning. With the learned model, we conduct real time personalized recommendation based on our novel efficient candidate generation approach. In addition, update recommendation is employed to interact with users according to different types of user feedbacks. Finally, we demonstrate the effectiveness of our system by both offline and online evaluation.**

***Keywords*—Paper Recommender System; Learning to Rank; Heterogeneous Academic Network; User Feedback**

I. INTRODUCTION

As the number of research papers available on the Web has increased enormously over the years, seeking the research papers of interest has become a crucial but burdensome task for researcher. Academic search engines (e.g., Google Scholar[1] and Microsoft Academic Search[2]) can help researchers search for papers based on specific keywords. However, search engine usually works well when users have some clear intents. They seldom provide personalized ranking according to users' specific interests, and cannot help users discover the most relevant and frontier work automatically. To address this problem, paper recommender system has been proposed to reduce the research paper seeking effort.

Paper recommender system provides personalized research paper recommendations to users according to his/her research interests. Specifically, in our work, paper recommendation aims to help users discover new and relevant research papers that he/she may would like to read. Note that it is quite different from the citation recommendation problem, which attempts to find papers that one may need to cite given a paper.

When designing an online paper recommender system, there are three basic problems we need to address. Firstly, how do we formalize and solve the personalized paper recommendation problem? McNee et al. [1] formalized paper recommendation as a rating prediction problem and applied collaborative filtering (CF) to solve it. However, recommendations are usually provided to users as a ranking list, which means the formalization of rating prediction will deviate from the ultimate target of this problem. Meanwhile, the cold start problem would be an inevitable issue in a CF system. Sugiyama et al. [2] took it as a ranking problem, and applied content-based filtering (CBF) to solve it. However, they only leveraged keyword based features and defined an ad hoc similarity function for recommendation.

Secondly, how does the system achieve real time recommendation? Both matrix factorization based and random walk based methods have been proposed for paper recommendation [3], [4]. These methods need to conduct matrix computation which is usually time consuming, and it would become even more challenging when the paper collection is extremely large and frequent online re-computation is needed when the matrix changes.

Finally, how does the system interact with users? There is few paper dealing with this problem in previous work, especially towards users' implicit or explicit negative feedbacks on recommendations. However, in practical recommender system this is a fundamental function that will largely affect user experience. Reasonable actions should be taken in response to different types of user feedbacks.

In this paper, we present a novel personalized paper recommender system, namely PaperTaste, to address these above issues. The recommender system is implemented as an key component of an online scholar platform SocialScholar[3], which is a vertical social network designed for computer science researchers. In PaperTaste, we formalize the personalized paper recommendation as a ranking problem with respect to users' research interests, and employ a supervised learning to rank (LTR) approach to solve the problem. Based on this formalization, we describe the design of our practical paper recommender system and its three most important components: i.e. offline training, online recommendation, and online feedback. In offline training, unlike most LTR methods relying on manually labeled training data which are both expensive and limited in size, we propose automatical training data construction by mining the existing large scale academic network. With the learned model, we conduct real time personalized recommendation based on our novel candidate generation ap-

[1]http://scholar.google.com
[2]http://academic.research.microsoft.com
[3]http://soscholar.com

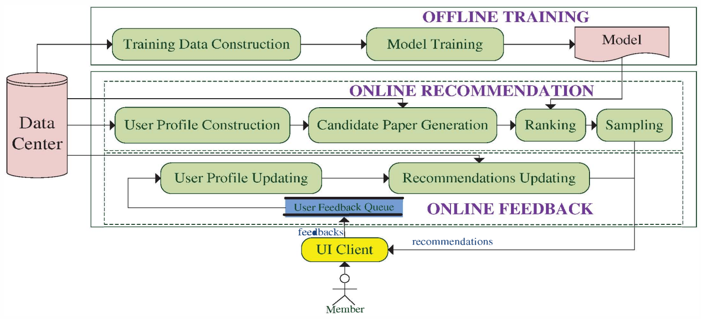


Fig. 1: The architecture of PaperTaste.

proach, and also provide updating recommendation according to different types of user feedbacks.

Our recommender system enjoys the following merits: 1) The formalization is directly towards the goal of the recommendation task since in most scenario recommended papers are represented as a ranking list; 2) By employing a learning to rank approach, we can elegantly involve various heterogeneous features to better capture users' research interests (as compared with conventional CBF methods), and also alleviate the cold start problem (as compared with CF methods); 3) The training is effective with large scale automatically constructed training data, and the prediction is efficient to meet the online requirements with our novel candidate generation approach; and 4) Our approach can well address both users' positive and negative feedbacks.

To verify the effectiveness of our recommender system, we conducted both offline and online evaluation. In offline evaluation, we compared our recommendation approach with some state-of-the-art baselines based on a dataset collected from SocialScholar, which contains $730,695$ papers and $10,000$ authors. In online evaluation, we analyze the performance of our recommender system based on online logs.

The rest of the paper is organized as follows: Section 2 reviews the related work. Section 3 provides the overview of our PaperTaste system. Section 4$\sim$6 describe the offline training, online recommendation, and online feedback, respectively. Section 7 presents the empirical experimental results. Finally, we conclude the paper in Section 8.

II. Related Work

Recommender systems provide a promising approach to seek information according to users' interests. There has been considerable research on recommendation approaches in the past decades. In this section, we will review some related work on recommender systems in the academic scenario.

Collaborative filtering [5], [6], [7] produces user specific recommendations of items based on patterns of ratings without need for exogenous information about either items or users [8]. McNee et al. [1] applied collaborative filtering to recommend citations. Their approach leveraged the citation web, a graph formed by the citations between research papers, to overcome the cold start problem. Yang et al. [9] proposed a ranking-oriented collaborative filtering approach based on users' access logs to avoid the problem of lacking user ratings. However, plenty of noises always exist in Web usage data, leading to the difficulty in extracting user profile accurately and fast. Tang and Mccalla [10] used artificial and human learners at the same time in an evolvable paper recommendation e-learning system, where artificial learners were used to solve the cold start problem when no paper had been rated by the learners.

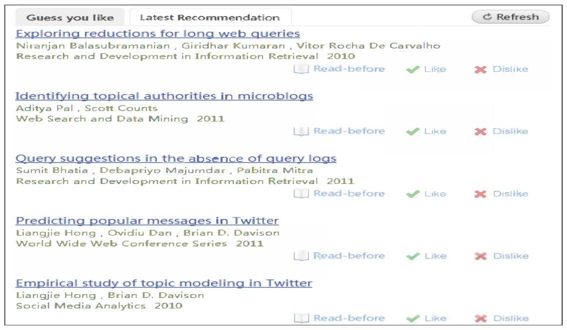
Beyond CF methods, content-based filtering methods have also been applied for academic recommendation. Sugiyama and Kan [2] used users' recent research interests to recommend new papers. They first constructed users' profiles using their previous publications, and then recommended papers by comparing the profiles with the contents of candidate papers. Gori and Pucci [4] proposed a research paper recommending algorithm based on the citation graph and random walk properties. The PaperRank algorithm is able to assign a preference score to a set of documents and link each other by bibliographic references. Liang et al. [11] incorporated various citation relations to recommend relevant papers. The method has two unique properties: Local Relation Strength measures the dependency between cited and citing papers, and Global Relation Strength captures the relevance between two papers in the whole citation graph.

Moreover, several studies proposed hybrid methods for academic recommendation. Huang et al. [12] propose a hybrid approach by representing books and users in an extended graph that incorporated book-to-book correlation, user-to-user correlation and book-to-user correlation. They employed classical graph search to extract and recommend useful information. Wang et al. [3] combined the merits of traditional collaborative filtering and probabilistic topic modeling. It provided an interpretable latent structure for users and items and produced recommendations for both existing and newly published articles. Torres et al. [13] proposed a hybrid method for recommending research papers by combining collaborative filtering and content-based filtering. They compared several methods ranging from only CF, only CBF and different combinations of CF and CBF.

III. Overview of PaperTaste

In this section, we first give an overview of our paper recommender system PaperTaste. The aim of the system is to provide personalized paper recommendations to users according to their research interests. As recommendations are usually provided to users as a ranking list, we formalize the recommendation problem as a ranking problem and propose employing a supervised learning to rank approach to solve this problem. The system architecture is shown in Figure 1. There are three major components in our system:

1. Offline Training: The goal of this offline component is to learn a recommendation model. There are two sub-modules to achieve this purpose. The Training Data Construction module mines the academic network to automatically construct training data and extracts various heterogeneous features for data representation; while the Model Training module conducts the optimization process to learn the ranking model. The learned model is then stored for online usage.
2. Online Recommendation: This component conducts online recommendation according to users' interests. Firstly, we extract users' profiles from their behaviors in our social scholar platform. Secondly, candidate



Exploring reductions for long web queries
Niranjan Balasubramanian , Giridhar Kumaran , Vitor Rocha De Carvalho
Research and Development in Information Retrieval 2010
Read-before Like Dislike

Identifying topical authorities in microblogs
Aditya Pal , Scott Counts
Web Search and Data Mining 2011
Read-before Like Dislike

Query suggestions in the absence of query logs
Sumit Bhatia , Debapriyo Majumdar , Pabitra Mitra
Research and Development in Information Retrieval 2011
Read-before Like Dislike

Predicting popular messages in Twitter
Liangjie Hong , Ovidiu Dan , Brian D. Davison
World Wide Web Conference Series 2011
Read-before Like Dislike

Empirical study of topic modeling in Twitter
Liangjie Hong , Brian D. Davison
Social Media Analytics 2010
Read-before Like Dislike

Fig. 2: The UI interface of PaperTaste.

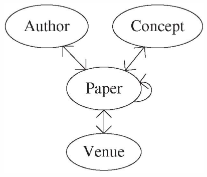


Fig. 3: The heterogeneous academic network.

papers are generated based on our novel candidate generation algorithm. These candidates are then ranked using the offline learned model. Finally, we cache the top recommendations and use a sampling strategy to generate dynamic paper recommendations to users.

3. Online Feedback: This component mainly focuses on handling user feedbacks. It receives different types of user feedbacks and takes corresponding actions in response, including updating user profile and then updating the paper recommendations accordingly.

There are also some other components which we do not list as major components but are of necessity in our recommender system. One is the Data Center component, which stores large scale data and provides a wealth of access interfaces for the recommender system. There are three major types of data stored in Data Center, i.e. academic data, user data, and logs. Academic data include different academic objects (e.g. research papers, authors, venues, and concepts) as well as the relations among these objects (e.g. authors write papers and papers publish in venues), which form a large scale heterogeneous academic network as shown in Figure 3. User data refer to the data generated by user behaviors in our social scholar system, such as user provided profile information, user actions (e.g. bookmark or comment on papers), user status, and so on. As these two types of data are mainly relational data, they are stored in the MySQL clusters. Log data refer to the information recorded by the system, such as page view, search and clicks. We leverage Scribe, an open source log server developed by Facebook, to collect the log data streamed in real-time.

Another component is the UI Client, which presents the recommendation results to users and interacts with users. The UI interface of PaperTaste is illustrated in Figure 2. As we can see, the recommended papers are shown in a list (including the paper title, authors, and published venue). For each recommendation, there are three types of actions users may take, i.e. Like, Dislike, and Read-before. On the top of the recommendation list, there are two tabs corresponding to two types of recommendations, i.e. “Guess you like” and “Latest Recommendation”. The difference between the two is that “Latest Recommendation” restricts that recommended research papers must be published in the latest year. Besides, users may also click the refresh button on the right-upper corner to obtain some new recommendations.

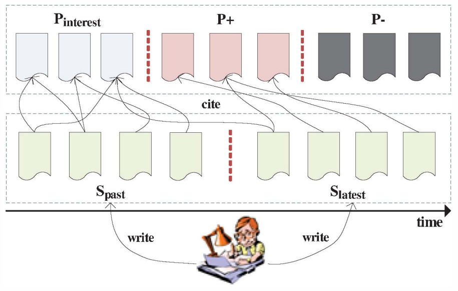


Fig. 4: The training data construction.

In the following sections, we will introduce the three major components in detail to show how PaperTaste works.

IV. Offline Training

In this section, we present the offline training component, which mainly aims to learn a recommendation model. In PaperTaste, we formalize the personalized paper recommendation as a ranking problem which is directly towards the task objective, and apply a supervised learning to rank approach to solve it. Specifically, we try to learn a model that can rank the candidate papers properly according to user’s research interest. Note that in PaperTaste user’s research interest is captured by a set of papers through his/her behaviors, which will be described in detail in Section 5.1. Therefore, the ranking task is actually to rank the candidate papers properly given a set of papers a user liked in the past.

To learn a supervised learning to rank model, the major problem is how we can obtain adequate labeled data for training. Obviously, there are no user generated data describing how users liked papers when the system was initially deployed. Meanwhile, it would be very difficult and expensive to label a large set of data manually for the task. Therefore, we propose a novel automatic approach to construct training data by mining the existing academic network and extracting various heterogeneous features (in Section 4.1). Based on the auto-constructed training data, we then learn a pair-wise ranking model for recommendation (in Section 4.2).

A. *Training Data Construction*

The training data for our problem, like that for conventional information retrieval (IR), consist of two objects in a structured way, i.e. user interest (as the query in IR) represented by a set of papers a user liked in the past, and papers a user will like/dislike in the future (as the relevant/irrelevant documents given the query). We propose to mine the existing large scale academic network to construct the training data automatically. The key idea comes from the observation that academic authors usually cite papers they have read and are really interested in. Therefore, we can view authors as users, papers cited by the author’s publications before a given time as the his/her interest, and those cited by the author’s publications after a given time as the his/her future interest. In this way, we can obtain a large scale dataset describing how people will like papers in the future given the papers they were interested in the past.

Specifically, we sample a set of authors from our academic network, and collect all the publications of these authors as well as their references. For each author, as illustrated in Figure 4, we separate his/her publications into two parts: One consists of publications in the latest year denoted as S_{latest}, and the other consists of the rest denoted as S_{past}. Papers cited by publications in S_{past} are viewed as the *interest profile* of the author, denoted by $P_{interest}$; While papers cited by those in S_{latest} but not within $P_{interest}$ represent the candidate interesting papers, denoted by P_{+}. We also randomly sample some other papers from the academic network to represent the candidate non-interesting papers, denoted by P_{-}. The whole candidate paper set is then denoted by $P_{candidate} = P_{+} \cup P_{-}$. We further associate multi-graded labels to each candidate paper automatically according to the citation frequency. Let l_i denote the label of the i-th paper $p_i \in P_{candidate}$, we have that

$$l_i = Freq(p_i, S_{latest})$$

where $Freq(p_i,S_{latest})$ denotes the number of papers in S_{latest} that cite the paper p_i. The basic assumption is that if a paper is more frequently cited by the authors' latest publications, it might be more interesting to the author and there is more chance that he/she will like it. Obviously, the papers from random sampling (i.e. P_{-}) would be assigned with the lowest label "0".

For each author, we further extract features for each candidate paper in $P_{candidate}$ given his/her interest profile $P_{interest}$. The features are used to describe how the candidate paper matches the author's interest. Unlike previous work which only used either similarity based on term vectors [2], [9] or citation relations between papers [1], [11], [4], we consider that there are various heterogeneous features which can be used for this description. Specifically, we define three types of features in our approach, i.e. *static features*, *content-based features*, and *relational features*, which are described briefly in the following.

Static Features Static features mainly describe the quality of the candidate paper, which is independent of interest profile. There are four static features for each candidate paper.

PaperRank is defined as the PageRank score of the paper computed over the paper citation network. This is a real valued feature within the range [0.15, +∞).

AuthorRank is defined as the PageRank score of the author computed over the author citation network. This is a real valued feature within the range [0.15, +∞).

VenueRank is defined as the PageRank score of the venue computed over the venue citation network. This is a real valued feature within the range [0.15, +∞).

Age is the number of lasting years from the publication year until now. This is a real valued feature within the range [0, 2013], since this year is 2013.

Content-based Features Content-based features capture the content similarity between the candidate paper and interest profile at different levels. In our work, we define three types of content-based features in total.

ContentSim is the maximum, minimum and average similarities on the title (TitleSim), abstract (AbstractSim), concept (ConceptSim), keyword (KeywordSim) and domain (DomainSim) between the candidate paper and interest profile. This is a real valued feature within the range [0, 1].

AuthorRadio is the percentage of papers in interest profile which contain some or all authors of the candidate paper. This is a real valued feature within the range [0, 1].

VenueRadio is the percentage of papers in interest profile which contain the venue of the candidate paper. This is a real valued feature within the range [0, 1].

Relational Features Relational features capture multiple types of relationships between the candidate paper and the interest profile. There are six relational features defined in our model.

Citation represents whether the candidate paper is a citation of interest profile. This is a nominal feature with a value either 0 or 1.

Reference represents whether the candidate paper is a reference of interest profile. This is a nominal feature with a value either 0 or 1.

Co-citation represents whether the candidate paper is a co-citation of interest profile. This is a nominal feature with a value either 0 or 1.

Co-reference represents whether the candidate paper is a co-reference of interest profile. This is a nominal feature with a value either 0 or 1.

Co-author represents whether the candidate paper is a co-author paper of interest profile. This is a nominal feature with a value either 0 or 1.

Co-venue represents whether the candidate paper is a co-venue paper of interest profile. This is a nominal feature with a value either 0 or 1.

B. Model Training

Based on the automatically constructed training data, we now learn a ranking model for recommendation. Learning to rank has been studied for years and a bunch of models have been developed, including pointwise, pairwise and listwise models [14], [15]. In our work, we employ the widely used pairwise learning to rank models for our problem. In pairwise models, preference object pairs are taken as instances in learning, and the ranking problem is formalized as classification of object pairs into two categories (correctly ranked and incorrectly ranked according to preference labels). In our work, we exploit Ranking SVM [16], which employs support vector machine as the classifier, for the learning task.

Specifically, for the t-th author, $t = 1, \ldots, m$, we construct preference paper pairs $(p_i^{(t)}, p_j^{(t)})$ with their labels satisfying $l_i^{(t)} > l_j^{(t)}$, representing that paper $p_i^{(t)}$ should rank higher than paper $p_j^{(t)}$. Ranking SVM can be formulated as the following optimization problem.

$$\begin{aligned} \min \quad & \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{t=1}^{m}\xi_t \\ s.t.: & (l_i^{(t)} - l_j^{(t)})(\boldsymbol{w}^T\boldsymbol{p}_i^{(t)} - \boldsymbol{w}^T\boldsymbol{p}_j^{(t)}) \geq 1 - \xi_t \\ & l_i^{(t)} > l_j^{(t)},\ \xi_t \geq 0,\ t = 1, 2, ..., m \end{aligned}$$

where $\boldsymbol{w}$ is the weight vector to be learned, $\boldsymbol{p}_i^{(t)}$ (or $\boldsymbol{p}_j^{(t)}$) denotes the feature vector of the i-th (or j-th) paper for the t-th author, C is the trade-off between training error and margin, and ξ is the slack variable. We employ gradient descent method for the optimization of the objective function. The learned model is then stored for online usage.

NOTE. As described above, we can see that essentially our approach falls into the CBF category, since we recommend papers based on user research interest representation and candidate paper representation. In this way, we can largely alleviate the severe cold start problem in CF methods. As compared with previous CBF methods, our approach extracts various heterogeneous features and elegantly involves them within a learning framework for better recommendation. It is believed that, when matching papers to user interests, many complicated features should be taken into account other than simple keywords from the paper content [2], [9].

V. Online Recommendation

This section introduces how we perform personalized paper recommendation in our online system in detail. Specifically, we first extract users' interest profile from their behaviors in our social scholar platform. Candidate papers are then generated based on both users' interest profile and our novel candidate generation algorithm. Finally, we rank the candidate papers using the offline learned model, cache the top recommendations and use a sampling strategy to generate dynamic paper recommendations to users.

A. User Profile Construction

In this step, we construct user interest profile from user behaviors in our SocialScholar platform. In SocialScholar, users can take different actions over papers, e.g. search, share or bookmark a paper. Currently, we deem the papers within the following explicit user behaviors as interested by the user and form the user's interest profile.

1) Papers declared by the user as his/her publications (denoted by P_D), and the references of these papers (denoted by P_R). The basic idea is that one's publications clearly represents his/her research interests, and those cited before are usually closed related to the interests.

2) Papers bookmarked by the user (denoted by P_B). Without loss of generality, users usually bookmark interesting papers for latter revisit.

3) Papers shared by the user (denoted by P_S). It is likely that one may share a paper to others that he/she thinks worth reading.

4) Papers liked by the user (denoted by P_L). Obviously, like is often a clear signal showing that the paper is within one's research interests.

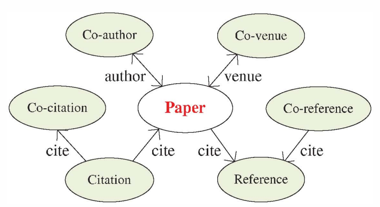


Fig. 5: The six paths to generate candidate papers.

5) Papers commented by the user (denoted by P_C). Typically, a user may comment a paper that he/she has read, which further indicates that the paper might be related to the user's research interests.

Note that we do not take the implicit user behaviors such as search, click or page view on some papers, since these weak signals may convey too much noise. In the future work, we will study how to take these behaviors into account to improve recommendation. The final user interest profile is the union of the papers in above behaviors.

$$P_{interest} = P_D \cup P_R \cup P_B \cup P_S \cup P_L \cup P_C$$

The benefit of considering multiple user behaviors is that as long as users have taken one of the above actions, he/she will be able to receive recommendations immediately.

B. Candidate Paper Generation

Based on the user's interest profile, we try to rank the papers beyond those in the profile using the learned ranking model for recommendation. However, since there are tens of millions of research papers in our academic network, this would make the ranking procedure prohibitively expensive. Therefore, we need a candidate paper generation process which can effectively prune the papers not interesting to users to form a relatively small subset of candidates. This is a critical step for the efficient online computation. One possible solution, like in conventional IR, is to use keyword matching for candidate generation. However, since there might be a set of papers in user's interest profile (i.e. a number of keywords), directly applying this method would generate too many candidates for computation. To overcome this problem, we propose a novel efficient generation process based on heterogeneous relations between papers.

This generation process comes from the observation how people find interesting papers to read in practice. Given some interesting papers already found, people may further seek papers to read by following the references and citations of those papers, or by following the authors and venues of those papers. These actions actually correspond to different paths between papers on the academic network. Therefore, given the interest profile of a user (i.e. a set of interesting papers), we propose to leverage the heterogeneous paths between papers to generate candidate papers. Basically, we use six types of paths between papers, i.e. "Citation", "Reference", "Co-citation", "Co-reference", "Co-author" and "Co-venue" as illustrated in Figure 5. The final candidate papers are the union of the papers generated by these six types of paths.

We analyze the effectiveness of our generation process based on our previously constructed dataset in offline component. Recall that for each author, we have the interest profile

Paths to generate candidates	Recall	Size
Citation	29.112%	2,071
Reference	24.067%	253
Co-citation	72.324%	12,050
Co-reference	37.481%	17,036
Co-author	33.164%	4,457
Co-venue	42.975%	55,099
Union	**81.706%**	90,052
keyword-based	**56.358%**	**658,957**

TABLE I: The evaluation of candidate generation process.

$P_{interest}$, and the candidate paper set $P_{candidate}$ including the interesting papers P_+ and randomly sampled non-interesting papers P_-. Therefore, an efficient candidate generation process should be able to generate a small set of candidates based on $P_{interest}$ that have a high recall of P_+ by pruning a huge number of non-interesting papers.

We compared the proposed path-based candidate generation process with keyword-based generation process in terms of recall and candidate size. As shown in Table I, when we use the six types of paths, the recall can achieve nearly 80%, which is reasonably good for the following recommendation step. But when we use keyword-based method, the recall is rarely 56%, and the size of candidate papers is extremely large.

C. Recommendation

The remaining task is to rank the candidate papers according to user interest profile based on the learned ranking model. Specifically, the ranking module extracts features for each candidate paper, computes its ranking score, and produces the ranking list according to the scores. We then cache the top K recommendations for efficiency (Note that in our online recommender system, K is set to 500).

Unlike in IR systems, users of a recommender system usually would not like to see a same ranking list multiple times. However, if we present results strictly according to their ranking scores in a descent order, users will always obtain the static recommendation results if no further behaviors he/she takes. Therefore, we adopt a sampling strategy to add some uncertainty and generate dynamic recommendation results to enhance user experience.

VI. Online Feedback

This component mainly focused on handling user feedbacks and updating the recommendations. As shown in the UI interface in Figure 2, there are three types of user feedbacks, i.e. Like, Dislike, and Read-Before. In the following, we will describe the corresponding actions in response to these different types of user feedbacks.

(1) **Like**. This action is a positive feedback, which reflects that the user is interested in this recommended paper. Then the updating procedure is conducted in two steps:

Firstly, we add this paper into the user's interest profile. Secondly, we generate candidate papers based on the paper user liked, merge them with those top K recommendations cached, rank the new candidate set according to the updated interest profile using the learned ranking model, and finally obtain the new top K recommendations.

(2) **Dislike**. This action is a negative feedback, which reflects that the user is not interested in this recommended paper. How to handle such negative feedback in recommendation has not been discussed much before. In this work, we take these non-interesting recommendations to form a black list for the user, as contrast to the user's interest profile mentioned before. This list has two functions: One is to ensure that those papers in this list would not be recommended to users any more; The other is to be used in updating the recommendations. The key idea is that those papers highly related to the black list should not be recommended in a high position. Therefore, the corresponding updating procedure is as follows:

Firstly, we add this paper into user's black list. Secondly, we generate the candidate non-interesting papers based on the black list in the similar way as the generation of candidate interesting papers based on the interest profile. If a candidate non-interesting paper hits the top K recommendations cached, that recommendation's score should be reduced. Specifically, we calculate the non-interesting ranking score for that paper based on user's black list using the same learned ranking model, and reduce the original recommendation score of the paper by the non-interesting ranking score.

(3) **Read-Before**. This action is a weak positive feedback, which reflects that the user has already read this recommended paper before. When users provide this feedback, there would be two options to take in the following, i.e. bookmarking this paper or not. If the user bookmark the recommendation, we consider that the user is interested in the paper (i.e. positive feedback) and thus we will take the updating procedure just the same as that of "Like". Otherwise, if the user decides not to bookmark it, it may indicate that the user is not interested in the paper any more (i.e. weak negative feedback), or that the user considers the paper related to his/her interest but not worth revisiting (i.e. weak positive feedback). Since this is an ambiguous weak signal, here we just add it to user's black list but do not trigger the recommendation updating process.

VII. Evaluation

In this section, we conduct both offline and online evaluations to demonstrate the effectiveness of our approach. For the offline evaluation, we conduct comparison between our recommendation approach with some state-of-the-art baselines on a dataset collected from SocialScholar. For the online evaluation, we analyze the performance of our recommender system based on online user logs.

A. Offline Evaluation

For the offline evaluation, an academic dataset is constructed from SocialScholar based on the method described in the offline training component. SocialScholar collected and combined papers from DBLP, IEEE, ACM and CiteSeer, and formed a large scale academic network consisting of $8,014,742$ papers, $4,432,205$ authors and $24,303,153$ citation relationships. For experiments, we randomly sampled $10,000$ authors who have published papers in the lastest two years (i.e. 2012-2013). For each author, three sets of papers were extracted from the academic network representing the interest profile $P_{interest}$, candidate interesting papers P_+ and candidate non-interesting papers P_-. The whole data set consists of $730,695$ papers.

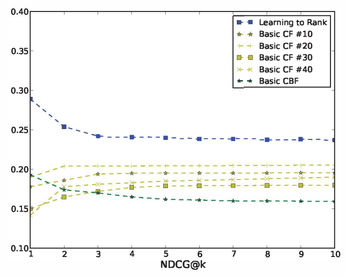


Fig. 6: Performance comparison between our approach and basic CF and CBF methods.

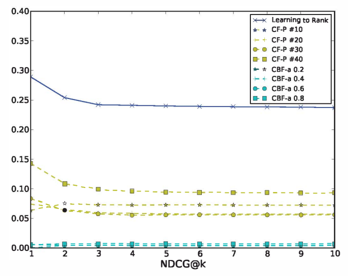


Fig. 7: Performance comparison between our approach and PageRank-weighted CF and CBF with α-PageRank.

We conduct empirical experiments by comparing our learning to rank based approach with two types of representative recommendation approaches:

(1) Basic CF: A typical user-based CF method is leveraged here. Specifically, for each test author, we find the most similar authors using k-nearest neighbor (kNN) algorithm, and then recommend the papers that the similar authors have been interested in but not within the test author's interest profile. Here we vary the number of nearest neighbor ranging from 10 to 40 in CF method.

(2) Basic CBF: In this method, only content-based information is leveraged to rank the candidate papers. Specifically, for each test author, based on his/her interest profile, we calculate the similarity between a candidate paper and the interest profile using the content information including title, abstract and concepts, and then provide the most similar papers as recommendation.

(3) PageRank-weighted CF: In this method, we use graph ranking to influence collaborative filtering. To integrate a graph ranking algorithm, we replace the user-unit-vector normalization with a normalization step which multiplies each paper's citation vector with the paper's importance score $r(u)$ such that $\hat{\mathbf{u}} = r(u)\mathbf{u}$. This causes papers with higher importance scores (e.g. higher PageRank) to exert more influence on the similarity of papers they cite, thus biasing the collaborative filter to favor that papers [17].

(4) CBF with α-PageRank: In this method, we generate score for each candidate paper using a linear combination of the similarity score and its PageRank within the whole citation web, using the following formula (where $L(i)$ is the similarity score and $r(i)$ is the PageRank)[17]:

$$s(i) = (1 - \alpha)L(i) + \alpha r(i)$$

For our model, we use 5-fold cross-validation, where three folds are regarded as training data, one fold as validation set and one fold as test data.

Since the recommendation is presented in a ranking list, we employ NDCG as the evaluation metric, which has been widely used in recent work on recommender system.

$$NDCG@k = \frac{1}{N_k}\sum_{i=1}^{k}\frac{2^{l_i} - 1}{\log(i+1)}$$

where l_i is the relevance label of the item with position i in the output ranked list, and N_k is a normalization constant.

Features	Weight
AuthorRatio	3.15
PaperRank	2.43
AuthorRank(min)	2.40
ConceptSim(max)	2.01
Age	1.38
AbstractSim(max)	0.74
TitleSim(max)	0.69
Co-author	0.69
VenueRatio	0.48
DomainSim(max)	0.37

TABLE II: The top 10 weighted features.

The performance comparison between our approach and baseline methods is shown in Figure 6 and Figure 7. From the results, we can see that the performance of CF methods mostly exceeds that of CBF method, although CF methods do not rank the top items very well. When varying the number of neighbors, the performance of CF methods reaches best when neighbor size is set to 20. It indicates that when neighbor size is too small, many interesting papers may not be found; while when neighbor size is too large, the performance may also be hurt mainly due to the data sparsity and more noisy neighbors involved. Moreover, our approach significantly outperforms the baseline methods (p-value$\prec$0.05). It indicates that by formalizing the recommendation task as a ranking problem and leverage various heterogenous features, our learning approach can well capture users' interests and provide better recommendations. We further analyze the importance of heterogeneous features, and list the top 10 weighted features in Table II. From the results we can see, all the three types of features show their effectiveness in capturing user interests, and among which AuthorRatio, PaperRank, and AuthorRank seem to be most important.

B. Online Evaluation

For the online evaluation, we analyze the user logs from our social scholar platform from February to April, 2013.

We first analyze the activation proportion and feedback proportion for our recommender system. The activation proportion refers to the proportion of users that activate PaperTaste among those login users of SocialScholar; while the feedback proportion refers to the proportion of users takes actions (i.e. feedback) over the recommendations. As shown in Table III, the average activation proportion over the three months is 14.09%, and among these users, more than 17%

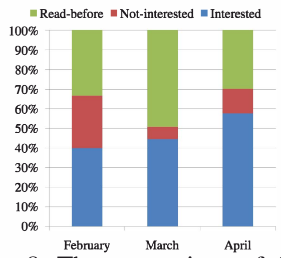


Fig. 8: The proportions of different feedbacks from February to April, 2013.

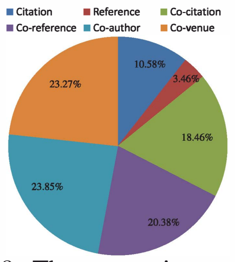


Fig. 9: The proportions of papers generated from six different paths in Interested feedback.

	Feb.	Mar.	Apr.	Overall
Activation Prop.	18.29%	17.79%	10.97%	14.09%
Feedback Prop.	11.67%	20%	17.21%	17.02%

TABLE III: The activation proportion and feedback proportion from February to April, 2013.

users provide feedback on the recommendation results. Over the three months, there is some decline of the activation proportion, and we find that it might be related to the boost of user registration and many new users may not immediately use PaperTaste. Meanwhile, the feedback proportion shows an increase tendency, which reflects more interactions between users and the recommender system.

We then further analyze the proportion of different feedbacks. As shown in Figure 8, the proportion of Interested feedback is the largest, which reaches 47.51% in average; while that of Not-Interested feedback is the least, which is 9.19% in average. This result shows the effectiveness of our recommender system in online scenario. Meanwhile, we can observe a clear increase tendency of the Interested feedback, from in February to in April. This might be related to the fact that when users take more actions in our system and form richer profiles, they would obtain better and better recommendations.

Finally, we focus on the interested feedback and analyze the effectiveness of different candidate generation paths. As shown in Figure 9, the most interesting papers come from the co-author and co-venue paths, indicating that users would like to follow the same author and venue to read papers. The least effective path is the reference path, which is also consistent with the recall results in Table I. The major reason might be related to the time factor. That is users are often more likely to find latest papers to read, while papers in reference are published earlier than what the users have already read.

VIII. Conclusion

In this paper, we describe the design of a practical paper recommender system PaperTaste. In our system, we take paper recommendation as a ranking problem, and solve it with a supervised learning to rank approach. We mine the existing academic network to automatically construct the training data, and extract various heterogeneous features for learning. We conduct real time personalized recommendation based on our novel candidate generation approach, and provide updating recommendation according to different types of user feedbacks. We demonstrate the effectiveness of our approach by both offline and online evaluations.

For the future work, we aim to take user implicit behaviors (i.e. search, click and view) into account to help form user interest profile and improve recommendation coverage and accuracy. With more and more users registering into our SocialScholar platform, we will also consider combining CF features to further enhance recommendation quality.

Acknowledgment

This research work was funded by the National Grand Fundamental Research 973 Program of China under Grant No. 2012CB316303, the National High-tech R&D Program of China under Grant No. 2014AA015204, the National Natural Science Foundation of China under Grant No. 61232010, Grant No. 61173064 and Grant No. 61203298, and the National Key Technology R&D Program of China under Grant No. 2012BAH39B04 and Grant No. 2012BAH39B02

References

[1] S. M. McNee, I. Albert, D. Cosley, P. Gopalkrishnan, S. K. Lam, A. M. Rashid, J. A. Konstan, and J. Riedl, "On the recommending of citations for research papers," in *CSCW '02*, 2002, pp. 116–125.

[2] K. Sugiyama and M.-Y. Kan, "Scholarly paper recommendation via user's recent research interests," in *JCDL '10*, 2010, pp. 29–38.

[3] C. Wang and D. M. Blei, "Collaborative topic modeling for recommending scientific articles," in *KDD '11*, 2011, pp. 448–456.

[4] M. Gori and A. Pucci, "Research paper recommender systems: A random-walk based approach," in *WI '06*, 2006, pp. 778–781.

[5] X. Su and T. M. Khoshgoftaar, "A survey of collaborative filtering techniques," *Advances in Artificial Intelligence*, vol. 2009, pp. 1–20, 2009.

[6] B. M. Sarwar, G. Karypis, J. A. Konstan, and J. Reidl, "Item-based collaborative filtering recommendation algorithms," in *WWW '01*, 2001, pp. 285–295.

[7] J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl, "An algorithmic framework for performing collaborative filtering," in *SIGIR '99*, 1999, pp. 230–237.

[8] F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, *Recommender Systems Handbook*, 2011.

[9] C. Yang, B. Wei, J. Wu, Y. Zhang, and L. Zhang, "Cares: A ranking-oriented cadal recommender system," in *JCDL '09*, 2009, pp. 203–212.

[10] T. Y. Tang and G. I. Mccalla, "Utilizing artificial learners to help overcome the cold-start problem in a pedagogically-oriented paper recommendation system," *Adaptive Hypermedia and Adaptive Web-Based Systems*, pp. 245–254, 2004.

[11] Y. Liang, Q. Li, and T. Qian, "Finding relevant papers based on citation relations," in *WAIM '11*, 2011, pp. 403–414.

[12] Z. Huang, W. Chung, T.-H. Ong, and H. Chen, "A graph-based recommender system for digital library," in *JCDL '02*, 2002, pp. 65–73.

[13] R. Torres, S. M. McNee, M. Abel, J. A. Konstan, and J. Riedl, "Enhancing digital libraries with techlens+," in *JCDL '04*, 2004, pp. 228–236.

[14] T. Qin, T.-Y. Liu, J. Xu, and H. Li, "Letor: A benchmark collection for research on learning to rank for information retrieval," *Information Retrieval*, 2009.

[15] Z. Cao, T. Qin, T.-Y. Liu, M.-F. Tsai, and H. Li, "Learning to rank: from pairwise approach to listwise approach," in *ICML '07*, 2007, pp. 129–136.

[16] T. Joachims, "Optimizing search engines using clickthrough data," in *KDD '02*, 2002, pp. 133–142.

[17] M. D. Ekstrand, P. Kannan, J. A. Stemper, J. T. Butler, J. A. Konstan, and J. T. Riedl, "Automatically building research reading lists," in *RecSys '10*, 2010, pp. 159–166.