Personalised Reranking of Paper Recommendations Using Paper Content and User Behavior

XINYI LI and YIFAN CHEN, University of Amsterdam, The Netherlands and National University of Defense Technology, China
BENJAMIN PETTIT, Elsevier, United Kingdom
MAARTEN DE RIJKE, University of Amsterdam, The Netherlands

Academic search engines have been widely used to access academic papers, where users' information needs are explicitly represented as search queries. Some modern recommender systems have taken one step further by predicting users' information needs without the presence of an explicit query. In this article, we examine an academic paper recommender that sends out paper recommendations in email newsletters, based on the users' browsing history on the academic search engine. Specifically, we look at users who regularly browse papers on the search engine, and we sign up for the recommendation newsletters for the first time. We address the task of reranking the recommendation candidates that are generated by a production system for such users.

We face the challenge that the users on whom we focus have not interacted with the recommender system before, which is a common scenario that every recommender system encounters when new users sign up. We propose an approach to reranking candidate recommendations that utilizes both paper content and user behavior. The approach is designed to suit the characteristics unique to our academic recommendation setting. For instance, content similarity measures can be used to find the closest match between candidate recommendations and the papers previously browsed by the user. To this end, we use a knowledge graph derived from paper metadata to compare entity similarities (papers, authors, and journals) in the embedding space. Since the users on whom we focus have no prior interactions with the recommender system, we propose a model to learn a mapping from users' browsed articles to user clicks on the recommendations. We combine both content and behavior into a hybrid reranking model that outperforms the production baseline significantly, providing a relative 13% increase in Mean Average Precision and 28% in Precision@1.

Moreover, we provide a detailed analysis of the model components, highlighting where the performance boost comes from. The obtained insights reveal useful components for the reranking process and can be generalized to other academic recommendation settings as well, such as the utility of graph embedding similarity. Also, recent papers browsed by users provide stronger evidence for recommendation than historical ones.

CCS Concepts: • **Information systems → Recommender systems**;

X. Li is now at the National University of Defense Technology, Changsha, China.

This research was partially supported by Ahold Delhaize, the China Scholarship Council, and the Innovation Center for Artificial Intelligence (ICAI). All content represents the opinion of the authors, which is not necessarily shared or endorsed by their respective employers and/or sponsors.

Authors' addresses: X. Li and Y. Chen, Science and Technology on Information Systems Engineering Laboratory, National University of Defense Technology, China; emails: lixinyimichael@gmail.com, y.chen4@uva.nl; B. Pettit, Elsevier, London, United Kingdom; email: b.pettit@elsevier.com; M. de Rijke, Informatics Institute, University of Amsterdam, Science Park 904, 1098 XH Amsterdam, The Netherlands; email: derijke@uva.nl.

1046-8188/2019/03-ART31 $15.00
https://doi.org/10.1145/3312528

Additional Key Words and Phrases: Academic search, paper recommendation, reranking

ACM Reference format:
Xinyi Li, Yifan Chen, Benjamin Pettit, and Maarten De Rijke. 2019. Personalised Reranking of Paper Recommendations Using Paper Content and User Behavior. *ACM Trans. Inf. Syst.* 37, 3, Article 31 (March 2019), 23 pages.
https://doi.org/10.1145/3312528

1 INTRODUCTION

Along with the digitization of academic resources and the increasing popularity of academic information platforms, online access to academic papers has become a widely used service. Various online academic service providers have given users access to papers through their search engines, such as Google Scholar [20], Aminer [66], and ScienceDirect [59], where users can enter queries to seek relevant papers in their database. In this scenario, users need to have an idea of what they are looking for, and the information needs can be formalized as queries. The search system takes a query as input and returns a ranking of relevant papers for users to examine and interact with.

While such academic search engines can often fulfill user requests by catering to specific information needs represented as queries, there are cases when users' information needs are not explicitly specified. For instance, users may want to learn about new developments in their domain by looking at emerging papers that are relevant. In this case, the user may not have an idea of what queries to enter on the search engine. This is a situation where paper recommender systems can step in and recommend relevant papers without the need for a user query.

Paper recommender systems have a role that is complementary to the search engine. The possible recommendation scenarios fall into three categories based on the recommendation timing:

(1) displaying paper recommendations before users start a new search session, based on their paper library or previously accessed papers [see, e.g., Google Scholar, 20];
(2) during a search session, displaying related recommendations beside the content that the user is currently browsing [see, e.g., ScienceDirect, 59]; and
(3) after a search session, sending emails of paper recommendations in the form of a newsletter [see, e.g., ScienceDirect, 59].

The first and third scenario fill the gap between user search sessions, while the second scenario is related to within-session recommendations.

In this study, we focus on the third scenario. We look at the ScienceDirect paper recommender, which sends a weekly email of paper recommendations to users. First, we provide a recommendation example from the system in Figure 1 to show how it works.[1] The recommender of ScienceDirect generates a ranked list of five paper recommendations based on the user's browsed papers. The email newsletter displays the title, venue (journal), authors, and publication date of each recommended paper. On clicking a recommendation, the user is linked to the paper on ScienceDirect. The system then logs on which recommendation(s) the user clicks. As a short summary, this system aims to recommend interesting papers to users based on their browsing history. A good recommendation list will place more relevant papers higher in the list.

Since the ScienceDirect paper recommender was released, an increasing number of users have signed up. It is especially challenging to make recommendations for these new users due to the lack of historical interactions with the recommender system. In this article, we address the challenge and try to come up with better recommendations for these new users. Specifically, we study the

[1] https://www.elsevier.com/connect/suffering-from-information-overload-personalized-recommendations-can-help.


ScienceDirect

Hello Daniel, here are personalized recommendations based on your latest signed in ScienceDirect activity.

Constructing a Natural Language Inference dataset using generative neural networks
Original research article
Computer Speech & Language, Volume 46, November 2017, Pages 94-112
Janez Starc, Dunja Mladenić

Beyond interviewer effects in the standardized measurement of ego-centric networks
Original research article
Social Networks, Volume 50, July 2017, Pages 70-82
Andreas Herz, Sören Petermann

Restricted Boltzmann machines for vector representation of speech in speaker recognition
Open access - Original research article
Computer Speech & Language, Volume 47, January 2018, Pages 16-29
Omid Ghahabi, Javier Hernando

Key players in conservation diffusion: Using social network analysis to identify critical injection points
Original research article
Biological Conservation, Volume 210, Part A, June 2017, Pages 222-232
Emmanuel K. Mbaru, Michele L. Barnes

More structural holes, more risk? Network structure and risk perception among marijuana growers
Original research article
Social Networks, Volume 51, October 2017, Pages 127-134
Aili Malm, Martin Bouchard, Tom Decorte, ... Marije Wouters

Fig. 1. An excerpt from a sample recommendation email sent to a ScienceDirect user based on his recent activity. The email contains five papers linked to ScienceDirect.

task of reranking the paper candidates generated by the current production system. Ranking is a very common module of the workflow in production recommendation systems, which usually include at least a candidate-generation phase and a ranking module [10, 13, 56]. The output of the system is generated by a multi-step process. We address this reranking task so that our model can easily be integrated into paper recommender systems (e.g., the ScienceDirect recommender). A direct application is to use our model to rerank the recommended papers generated by the ScienceDirect recommender system.

Over 14 million papers are indexed on ScienceDirect [60]. Picking the few papers that may appeal to the user is not a trivial task. Collaborative filtering techniques are often used in recommender systems to generate a candidate pool of papers based on user-paper interactions. Even though there was initially no data on user interaction with the recommender system, there was still a wealth of data on user interactions with papers on ScienceDirect. Apart from this behavioral aspect, paper metadata may also assist the recommendation task by providing similarity measures that are based on paper contents, e.g., to recommend semantically similar papers, or papers that are authored by the same or similar authors.

In this article, we propose a hybrid model that combines content and behavior to rerank the candidate paper recommendations generated by the ScienceDirect recommender. First, we propose several content-based measures that are derived from various paper aspects, such as word space similarity, and author similarity from an embedding space. Next, we use joint matrix factorization to learn a mapping from a user's browsed articles on the search engine to a user's clicks on the recommendations, to alleviate the sparsity of the recommendation click data. We use a pairwise learning model to rerank the candidate paper recommendation, which eventually leads to better results in offline evaluations based on real email click data.

The contribution of this article mainly lies in: (1) "task transfer" for the academic setting: data collected for one task (search) is used to help optimize performance on another (recommendation), and (2) how to combine content and user behavior to generate high-quality academic recommendations. The framework captures user interests on different paper aspects, as well as alleviating the sparsity problem in click data. The recommendation framework for ScienceDirect has implications for academic recommendation settings that share similar inputs. The framework relies on two kinds of input: paper properties and user interactions. The paper properties that we have utilized can be found on many popular academic search engines such as Google Scholar [20], Semantic Scholar [61], and CiteseerX [35]; the user interactions, i.e., how users interact with the search engine and the recommender, respectively, are also available. For instance, in Google Scholar's search interface, there is a snippet showing recommended papers below the search bar for users to click on, even when users have not entered a search query; Semantic Scholar also has its proprietary email alert service that can send relevant paper recommendations.

The article is structured as follows. We describe the models that we propose in Section 2, the experimental setup in Section 3, and the results and analysis in Section 4. We present related work in Section 5 and conclude in Section 6.

2 MODELS

In this section, we introduce the models for the paper recommendation task. First, we introduce the production baseline, because it provides the candidates for our proposed reranking model. Then, we introduce our hybrid reranking model (*Hybrid Reranking Model* (HRM)) that considers both behavior and content and reranks the candidates. Here, "hybrid" refers to using both content and behavior.

2.1 Production Baseline

The production system takes the paper browsing history of a user as input and produces a ranked list of five paper recommendations. While we are not able to elaborate on the exact details of the production system, we can describe the core part of the algorithm: The five candidates are generated and ranked by an algorithm that uses an item-item neighborhood-based collaborative filtering method [41, 57], based on usage similarity from ScienceDirect browsing logs. We refer to this paper-paper similarity as *browsing similarity* in the remainder of the article.

In this study, we apply the reranking model to the top five candidates from the production system, and compare the model's ranking to the production baseline. The top five candidates were chosen because for these recommendations there is email click feedback that enables offline evaluation; if successful, the model could be applied to a longer list of candidates.

2.2 Proposed Model

In this section, we introduce the *Hybrid Reranking Model* (HRM) by first providing an overview of the model architecture and then delving into the details. The model scores paper recommendation candidates generated by the production system, using both content and behavior components, which will be explained shortly. The candidates are then reranked by the score.

2.2.1 Model architecture overview. An overview of the model is shown in Figure 2. A two-layer feedforward neural network is used as the scoring function, where the input layer takes features from each candidate paper, and the output layer contains one node that yields the score.

We explain what the input feature representations are in Figure 2 from left to right: S_{recent} and $S_{history}$ are the similarity between recommendation candidates and users' browsed papers. They contain the average similarity scores of each paper aspect by comparing the candidate

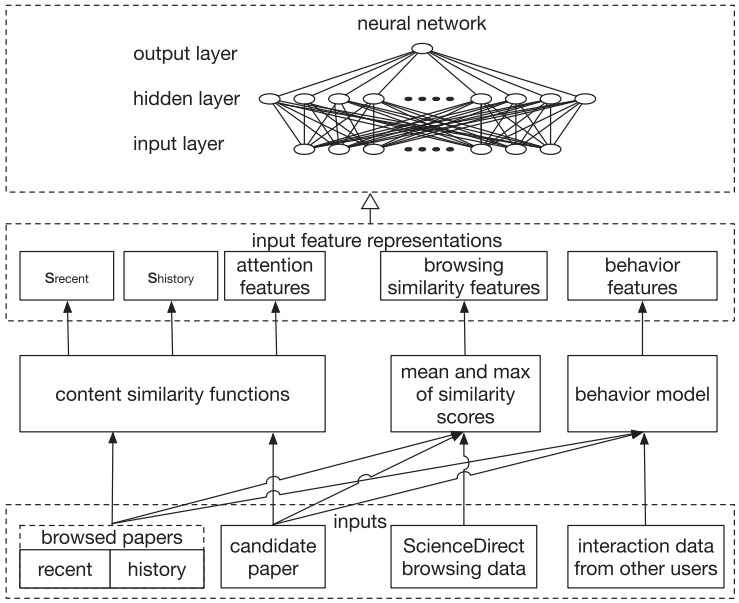

Fig. 2. Architecture of the Hybrid Reranking Model (HRM) that shows how a candidate paper recommendation for a user is scored.

paper against the recent papers and historic papers, respectively; the attention features on different fields of papers and on recent/historical papers are derived from a user's browsed papers. These functions will be described in Section 2.2.4.

The browsing similarity features used by the production system are based on ScienceDirect browsing data: We use the mean and maximum similarity scores of each paper recommendation candidate compared against papers in the browsing history. The behavior features are the predicted click scores from the behavior model to be described in Section 2.2.5. Together these features determine the inputs for Hybrid Reranking Model (HRM).

Training is done by optimizing a pairwise hinge loss from the preferences of clicked papers $\mathcal{R}_u^+$ over the non-clicked papers $\mathcal{R}_u^-$ for each user u, as follows:

$$L(\mathcal{R}_u^+, \mathcal{R}_u^-) = \sum_{p_i \in \mathcal{R}_u^+} \sum_{p_j \in \mathcal{R}_u^-} \left[1 - f(x_{p_i}) + f(x_{p_j})\right]_+, \tag{1}$$

where $f(\cdot)$ denotes the scoring function (neural network), and x_{p_i} and x_{p_j} denote the feature representations for clicked paper p_i and non-clicked paper p_j, respectively.

We apply rectified linear unit (ReLU) activations on the hidden layer for efficient learning. We apply linear activations on the output layer, because this ensures an unbounded value for the pairwise loss function and also performs best in our experiments. We use the Adam optimizer [32] and mini batches during training.

Next, we introduce how we consider paper metadata to measure different types of similarity. Below, we provide formal representations of various paper aspects, of users, and then the similarity functions for them.

Before continuing, let us briefly introduce the main notation that we will be using in the remainder of the article; see Table 1.

Table 1. Notations Used in the Article

Notation	Description
p, p_i, p_j	papers
$A(p)$	set of authors of paper p
$V(p)$	venue where paper p is published
$F(p)$	freshness score of paper p
$I(p)$	impact score of paper p
$P(p)$	popularity score of paper p
$W(p)$	word space representation of paper p
$E(p)$	entity space representation of paper p
$SimX$	similarity of field X for 7 choices of X (V, A, F, I, P, W, E)
α_{field_i}	attention feature for $field_i$ from 7 fields (V, A, F, I, P, W, E)
α_{recent}	attention score on the user's recent papers
$\alpha_{history}$	attention score on the user's historic papers
m	number of users
n	number of papers
k	number of latent dimensions
$B \in \mathbb{R}^{m \times n}$	paper browsing history matrix
$R \in \mathbb{R}^{m \times n}$	paper click history matrix
$S \in \mathbb{R}^{n \times n}$	paper-paper browsing similarity matrix
$D \in \mathbb{R}^{n \times n}$	a diagonal matrix whose entries are the row sums of S
$M \in \mathbb{R}^{n \times n}$	matrix to map browse to click
$Q \in \mathbb{R}^{n \times k}$	paper factor matrix
$\mathrm{Tr}(\cdot)$	the trace operator of a matrix
$s_{ij} \in \mathbb{R}^{n \times n}$	browsing similarity between paper p_i and p_j
$\boldsymbol{q}_i$	latent factor for paper p_i
b_i	bias of paper p_i
r_{ui}	predicted score of click of user u to paper p_i
$\mathcal{B}_u^+$	set of papers browsed by user u
$\mathcal{R}_u^+$	set of papers clicked by user u
$\mathcal{R}_u^-$	set of papers shown to but not clicked by user u
$\mathcal{R}_u$	set of papers in the candidate, i.e., $\mathcal{R}_u = \mathcal{R}_u^+ \cup \mathcal{R}_u^-$

2.2.2 Paper Representations. Each paper p is represented as a collection of different aspects, grouped as follows:

- metadata from papers: author A, venue V, freshness F, word space W, entity space E;
- metadata from user interactions: impact I and popularity P.

These aspects are available for all papers and users in our scenario and considered to be potentially useful for the recommender system. The reason for considering authors and venues is that users may be interested in papers from the same or similar authors, and those published in the same or similar venues. The word space and entity space measure content similarity and are thus also potentially useful. Besides, academic searchers may seek papers with high impact when they are learning about a domain, or popular papers (e.g., those with many downloads) that their community is discussing, or newly published papers that track the latest developments, hence the inclusion of impact, popularity and freshness.

Formally, we can think of every paper p as a tuple $p = \langle A(p), V(p), F(p), W(p), E(p), I(p), P(p)\rangle$, where each aspect is defined as follows:

Authors:

$$A(p) = [a_1, a_2, \ldots, a_n], \tag{2}$$

where a_i is an author of the paper p.

Venue:

$$V(p) = v_i, \tag{3}$$

meaning that paper p is published in venue (journal) v_i.

Freshness Score: We also model how "fresh" a paper p is, defined as

$$F(p) = \frac{1}{e^{t_{current} - t_{publish}(p)}}, \tag{4}$$

where $t_{current}$ is the current time, and $t_{publish}(p)$ is the time of the paper p being published online. $F(p) \in (0, 1]$. The more recently a paper has been published, the higher the freshness score is.

Impact Score: We use citations as a measure of impact for papers, which is defined as

$$I(p) = \frac{\log(c(p) + 1)}{\log(c_{max} + 1)}, \tag{5}$$

where $c(p)$ is the citation count of paper p, and c_{max} is the maximum number of citations in the dataset.

Popularity Score: The popularity of a paper p reflects how often users interact with the paper. We use the number of downloads to represent popularity:

$$P(p) = \frac{\log(d(p) + 1)}{\log(d_{max} + 1)}, \tag{6}$$

where $d(p)$ is the number of downloads of paper p, and d_{max} is the maximum number of downloads of a paper in the dataset.

Word Space: To represent a paper p in a word space $W(p)$, we use tf-idf vectors, with values for words and bigrams in the article title, abstract and keywords. We remove English stop words, very common words and very rare words before calculating the tf-idf values. In the end, each paper is represented as a sparse vector of size of 2^{21}, with hashing to determine token indices in the vector.

Entity Space: While word space measures such as tf-idf similarities can be used to directly compare the contents of papers, an entity space representation $E(p)$ is able to provide us with additional information that incorporates both structure and semantics through graph embeddings [3, 40].

We first build a knowledge graph by using important aspects of a paper including keyword, author and venue. The graph contains four node types, paper, author, keyword, and venue nodes, and three relations (predicates) between a paper and an aspect as listed below:

- hasAuthor: the paper has this author;
- hasKeyword: the paper contains this keyword; and
- publishedInVenue: the paper is published in this venue (journal).

Next, to compute the entity space, we use the TransE model [3] to derive embeddings based on knowledge graphs. As input the model takes the triplets in the graph; these have the form (h, r, t), with a head entity h, a relation (predicate) r, and a tail entity t. The objective of the model is to learn embeddings so that $h + r$ lies in the proximate neighborhood of t if such a triplet (h, r, t)

exists in the training set, and $h + r$ will be far away from t if the triplet is not valid. The model learns embeddings by minimizing a pairwise hinge loss:

$$\sum_{(h,r,t)\in T} \sum_{(h',r,t')\notin T} [1 + \|h + r - t\|_2 - \|h' + r - t'\|_2]_+, \tag{7}$$

where T denotes the training set of triples. Negative triplets (h', r, t') are sampled by replacing either the head or the tail entity with another random entity. After training, the cosine distance of the node embeddings reflects their proximity in the knowledge graph.

Due to the relatively high computational costs of working with knowledge graphs [3], we derive the embeddings on a subgraph instead of on the complete graph. We choose a reasonable size for the subgraph so that it is computationally feasible and also alleviates the sparsity problem in the node connections. The subgraph is comprised of the union of the browsed papers and recommended papers from 65,994 users, a superset that is about 15 times the size of the set of users that we will study in our experiments. In total, we have 609,716 paper nodes, 1,650,470 author nodes, 3,961 venue nodes and 808,845 keyword nodes, plus 6,103,728 relation edges.

The graph is then used as input for the TransE model to derive embeddings of the nodes in the graph. In the end, we obtain embeddings for papers, authors and venues. These embeddings will be used later in content similarity measures.

2.2.3 User Representations. The user representations are straightforward: each user u is represented as a collection of papers in their browsing history:

$$u = [P_{recent}, P_{history}], \tag{8}$$

$$P_{recent} = [p_1, p_2, \ldots, p_k], \tag{9}$$

$$P_{history} = [p_{k+1}, p_{k+2}, \ldots, p_n]. \tag{10}$$

We segment a user's browsed papers into two sets, the recent ones, P_{recent}, and the historical ones, $P_{history}$. We write p_i to refer to the ith paper in each of the segmentations, in the order of occurrence in the user's timeline starting from the most recent one. In academic search, users' topic interests may shift over time [37]. We make this segmentation so that it may help us compare the user's recent interests against their historical interests, and see whether and to which extent there is a deviation.

In case of a large deviation, P_{recent} should provide more support to generate paper recommendations.

Specifically, the clicked papers in the most recent session are put into P_{recent} if it contains at least clicks on two different papers, and the rest into $P_{history}$. Otherwise, we select the most recent θ papers from u into P_{recent} and put the rest into $P_{history}$. Papers in P_{recent} and $P_{history}$ are deduplicated.

2.2.4 Content Similarities. Based on the user and paper representations, in this section, we describe similarity functions to measure different types of content similarity. Specifically, the content component measures the similarity between candidate recommendations and users' browsed papers using information from the paper metadata. The output consists of similarity scores to feed into the reranking model.

Field-level Similarities and Attention Features. First, we introduce similarity measures for individual fields, which are used to compare paper similarities in each field. When comparing two papers p_i and p_j, the similarity of each field is defined as follows.

For the word space and entity space, we use the cosine similarity of the vectors that represent each paper. The cosine similarity between two vectors v and v' is defined as

$$\cos(v, v') = \frac{v \cdot v'}{\|v\| \cdot \|v'\|}, \tag{11}$$

where the similarity value $\cos(v, v')$ ranges between -1 and 1.

Then, the similarities for word and entity space are

$$SimW(p_i, p_j) = \cos(W_{p_i}, W_{p_j}), \tag{12}$$

$$SimE(p_i, p_j) = \cos(E_{p_i}, E_{p_j}), \tag{13}$$

where W_{p_i} is the tf-idf vector and E_{p_i} is the paper entity vector for paper p_i obtained from the output of the TransE model [3].

Similarly, a venue entity vector $E_{v_{p_i}}$ for paper p_i and an author entity vector E_{a_m} for author a_m of p_i are obtained from the output of the TransE model [3]. We apply a "soft match" approach when comparing venue and author similarities. Compared to an "exact match" approach where the similarity ends up being either 1 (same) or 0 (different), the "soft match" approach outputs a continuous similarity score. For instance, "Accident Analysis & Prevention" and "Safety Science" being two different journals (with no overlapping terms in the journal title), they would have a similarity score of 0 in the "exact match" approach. However, in the embedding space they would have a similarity score of 0.48, representing a more precise estimate of the inherent similarity.

Then, venue- and author-based similarity measures, $SimV(\cdot, \cdot)$ and $SimA(\cdot, \cdot)$ are defined as follows:

$$SimV(p_i, p_j) = \cos(E_{v_{p_i}}, E_{v_{p_j}}), \tag{14}$$

$$SimA(p_i, p_j) = \begin{cases} \dfrac{\sum_{a_m \in A_{p_i}} \max_{a_n \in A_{p_j}} \cos(E_{a_m}, E_{a_n})}{|A_{p_i}|}, & \text{if } |A_{p_i}| \leq |A_{p_j}| \\ \dfrac{\sum_{a_n \in A_{p_j}} \max_{a_m \in A_{p_i}} \cos(E_{a_n}, E_{a_m})}{|A_{p_j}|}, & \text{otherwise,} \end{cases} \tag{15}$$

where v_{p_i} is the venue of paper p_i, $E_{v_{p_i}}$ is the corresponding paper entity vector; A_{p_i} is the set of authors of paper p_i, and E_{a_m} is the entity vector for author a_m. Note that in the author similarity function, we examine each author from the smaller author set and find the most similar one in the other set and then calculate the average of the similarities. This ensures that $SimA(p_i, p_j)$ is symmetrical.

For freshness, impact, and popularity, these three measures are single value features. We use $L1$ distance with an adjusted weighting to obtain their similarities:

$$SimF(p_i, p_j) = (1 - \|F(p_i) - F(p_j)\|_1) \times \max(F(p_i), F(p_j)), \tag{16}$$

$$SimI(p_i, p_j) = (1 - \|I(p_i) - I(p_j)\|_1) \times \max(I(p_i), I(p_j)), \tag{17}$$

$$SimP(p_i, p_j) = (1 - \|P(p_i) - P(p_j)\|_1) \times \max(P(p_i), P(p_j)). \tag{18}$$

We define the weighting to capture the similarities only when two papers both have a high value in this field. In cases where both have a low value, the similarity value will be "down-weighted," representing a weaker level of evidence for similarity. For instance, given two paper pairs with low impact values (0.1, 0.2) and high impact values (0.8, 0.9), the similarity score would be 0.09 and

0.81, respectively. Although the absolute difference of impact is the same for both pairs (0.1), the pair with relatively high values has a much larger similarity score.

Field Level Attention. Now that we know how to obtain the similarity scores *SimX* for seven choices of X (V, A, F, I, P, W, E), we would like to further know which specific fields the user may be focusing on while browsing papers. This is to tailor the recommendations for those fields, be it the semantic similarity, venues or authors. These "attention features" are implicit. However, we can derive the attention features through past user interactions. In particular, we assume that they can be inferred from P_{recent} (users' recently browsed papers). We hypothesize that for a set of papers, if the average pairwise similarities of certain aspects are higher than other fields, it is probably because users are paying attention to these aspects. For instance, high word space similarity indicates that users are sticking to a specific topic. Likewise, if the venue and freshness similarity scores are high, this could be that the user is mostly checking papers that are both recent and are from a specific journal. We use the averaged pairwise similarities calculated by each field as the field-level attention feature.

The attention feature for $field_i$ is the sum of its pairwise similarities divided by the number of paper pairs in P_{recent}:

$$\alpha_{field_i} = \frac{\sum_{p_i, p_j \in P_{recent}, i \neq j} Sim_{field_i}(p_i, p_j)}{C^2_{|P_{recent}|}}, \tag{19}$$

where $C^2_{|P_{recent}|}$ refers to the number of paper pairs.

Recent and History Attention. The users' recent and historical paper interactions may both provide evidence to surface good recommendations. We make the distinction between recent and historical papers, because users' interests may evolve over time. When the users' recent interests are significantly different from their historical interests, the recommender should be aware of this deviation. Therefore, we define attention features for this situation, where α_{recent} and $\alpha_{history}$ represent the two attention scores on the user's recent and historic papers.

α_{recent} and $\alpha_{history}$ are calculated using the browsed papers (P_{recent} and $P_{history}$). The more the user's recent interests deviate from the historic interests, the higher the value of α_{recent}, hence providing a bias feature to consider the more recent user activities. It is calculated as follows:

$$\alpha_{recent} = Distance(P_{recent}, P_{history}), \tag{20}$$

$$\alpha_{history} = 1 - \alpha_{recent}. \tag{21}$$

The distance $Distance(P_{recent}, P_{history})$ is calculated by averaging over the distance of each paper in P_{recent} to its closest match in $P_{history}$. The idea of finding each paper's closest match instead of averaging over all papers in $P_{history}$ is because the history may be diversified: A recent paper may be very similar to one paper in the history but different from the rest. In case there is at least one similar paper in $P_{history}$, we consider that the current paper being examined does not deviate far from the history. Formally, the distance is defined as follows:

$$Distance(P_{recent}, P_{history}) = \frac{\sum_{p_i \in P_{recent}} \min_{p_j \in P_{history}} (1 - \cos(W_{p_i}, W_{p_j}))}{|P_{recent}|}, \tag{22}$$

where $1 - \cos(W_{p_i}, W_{p_j})$ is the cosine distance between two papers' tf-idf vectors.

So far, we have explained how we exploit the content aspects for recommendation that are based on the paper metadata. Next, we introduce the behavior aspect where user-paper interactions are concerned.

2.2.5 Behavior. Paper metadata provides evidence for recommendations from the content perspective. User interactions, i.e., users' browsing behavior on the search engine and clicks on recommendation emails, also provide signals for generating good recommendations. In our scenario, the users have past browsing behavior but no clicks prior to their first interaction with the recommender system.

Nevertheless, the paper-paper browsing similarities are available to us (as used by the production system, described in Section 2.1). They provide a measure of behavior-based similarity based on readership of all users on ScienceDirect.[2] Naturally, we can incorporate this external similarity information into our model.

We devise a behavioral model, that utilizes both browsing and click behavior in the interaction log. The motivations of our model are given below:

- For new users, there are no prior email clicks for predicting their interactions with the recommender. To address the issue, we complement the absence of click ratings by using the browsing history. Obviously, browsing papers on the search engine and clicking a paper in the email are two different user interactions with papers. Thus, a mapping function is required to transform browsed papers to email clicks. It is not possible to learn the mapping for every user as there may be no click at the time of recommendation, but it is possible to utilize the browsed papers and email clicks of other users (and this data quantity will grow over time). Essentially, we try to infer the clicks of new users from other users' mappings, using supervised learning.
- As paper recommendations are shown in a relatively compact email, we assume that users have noticed all the papers. Therefore, a user's clicks on the five shown papers in the email entail implicit pairwise preferences. For instance, given five papers p_1, p_2, p_3, p_4, and p_5, if the user clicks paper p_2 and p_3 in the list of five papers, then it is reasonable to assume that they prefer paper p_2 and p_3 over paper p_1, p_4, and p_5.
- The paper-paper similarity based on a user's browsing history is available. It is likely more accurate than the similarity from user clicks in emails, because it is based on the complete set of ScienceDirect users, which is several orders of magnitudes larger. Moreover, it captures transitive similarities from a global perspective. Therefore, it is important for our model to preserve this similarity.

Recall that our notation was introduced in Table 1; it is used in the following model descriptions. We propose to learn a mapping function from user browsed papers to user clicks on the email, denoted as

$$R \sim BM, \tag{23}$$

where $B, R \in \mathbb{R}^{m \times n}$ are the matrices for browses and clicks, respectively, and $M \in \mathbb{R}^{n \times n}$ is a mapping matrix. In practice, n is generally very large so that it could pose a great burden to learn M. Thus, we propose to factorize M into the multiplication of a low-dimensional paper factor $Q \in \mathbb{R}^{n \times k}$, as follows:

$$M \sim QQ^T, \tag{24}$$

where T is the transpose operator of a matrix.

Based on the assumptions given in Equations (23) and (24), we can predict the click of user u on paper p_i by the following equation:

$$\tilde{r}_{ui} = b_i + \boldsymbol{q}_i^T \sum_{t \in \mathcal{B}_u^+} \boldsymbol{q}_t, \tag{25}$$

[2]The involved users outnumber the users we study in our experiments by several orders of magnitude.

where b_i is a scalar for the bias of paper p_i. $\mathcal{B}_u^+$ is the set of papers browsed by user u. Here, we ignore the user bias in Equation (25), since it is unknown for new users. Note that we do not exclude p_i from $\mathcal{B}_u^+$, as suggested by the item-based collaborative filtering methods. This is because the set of candidate papers does not overlap with the set of browsed papers, i.e., $\mathcal{B}_u^+ \cap C_u = \emptyset$. Each time we draw a pair of papers (p_i, p_j) for each user to learn Q and b_i, b_j, where $p_i \in \mathcal{R}_u^+$ and $p_j \in \mathcal{R}_u^-$, and optimize a pairwise loss function given by Bayesian personalized ranking [54]:

$$\mathcal{L}(u, p_i, p_j) = -\log \sigma\left(\tilde{r}_{ui} - \tilde{r}_{uj}\right), \tag{26}$$

where $\sigma(\cdot)$ stands for the sigmoid function. To preserve paper-paper similarities from the browsing history, we follow the assumption that the distance between $\boldsymbol{q}_i$ and $\boldsymbol{q}_j$ is small when s_{ij} is large. Without loss of generality, we adopt the Euclidean distance, e.g., $\|\boldsymbol{q}_i - \boldsymbol{q}_j\|_2^2$. We can then define the following similarity regularization terms:

$$\begin{aligned}\frac{1}{2}\sum_{i,j}^{n}\|\boldsymbol{q}_i - \boldsymbol{q}_j\|_2^2 s_{ij} &= \sum_{i=1}^{n}\boldsymbol{q}_i^T\boldsymbol{q}_i d_{ii} - \sum_{i,j}^{n}\boldsymbol{q}_i^T\boldsymbol{q}_j s_{ij} \\ &= \mathrm{Tr}\left(Q^T D Q\right) - \mathrm{Tr}\left(Q^T S Q\right) = \mathrm{Tr}\left(Q^T L Q\right),\end{aligned} \tag{27}$$

where $\mathrm{Tr}(\cdot)$ is the trace operator of a matrix, D is a diagonal matrix whose entries are the row sums of the browsing similarity matrix S (S is symmetric), i.e., $d_{ii} = \sum_{j=1}^{n} s_{ij}$, and $L = D - S$ is the Laplacian matrix of the graph [9]. Putting Equations (26) and (27) together, the model is given as follows:

$$\min_{Q, \{b_i\}} \sum_{u=1}^{m} \sum_{\substack{p_i \in \mathcal{R}_u^+, \\ p_j \in \mathcal{R}_u^-}} -\log \sigma\left(\tilde{r}_{ui} - \tilde{r}_{uj}\right) + \alpha \mathrm{Tr}\left(Q^T L Q\right) + \frac{\lambda}{2}\|Q\|_F^2. \tag{28}$$

The first term in the objective function captures the pairwise preferences of every user over the papers shown in the emails. The second term preserves the paper-paper similarities in the browsing history through graph regularization. Graph regularization is widely used to preserve similarities, e.g., social regularization [43] and locality regularization [58]. The third term regularizes Q to avoid overfitting. α and λ are hyper-parameters.

We optimize Equation (28) via Stochastic Gradient Descent. The optimizing procedure is similar to Reference [18], i.e., for each user u, we sample a positive item $i \in \mathcal{R}_u^+$ and a negative item $j \in \mathcal{R}_u^-$, and optimize Equation (28) with respect to (u, i, j). Note that we train Equation (28) first, and then train Hybrid Reranking Model (HRM) (Equation (1)) given the output scores generated by the behavior-based model. While we can devise an end-to-end model to train the behavior-based model jointly with HRM, the optimization procedure can be very complex and inefficient. This is because the training procedures of Equation (28) and HRM are very different. The behavior-based model is trained via sampling to capture the implicit relationships between items, whereas HRM assumes the independency among inputs to perform mini-batch training.

3 EXPERIMENTS

In this section, we describe the experiments, including research questions, data preparation, and experimental setup.

3.1 Research Questions

We aim to find out how to utilize both content and behavior to rerank paper recommendations. We are interested in whether HRM, which utilizes content and behavior, can beat the production baseline, and how useful the different input features are. Specifically, we answer the following research questions.

(1) Does HRM, which utilizes content and behavior, provide improvements in reranking over the production baseline?
(2) What is the utility of the content features and behavior features in reranking, respectively?
(3) Within the content features, for paper similarity based on various paper aspects, which paper aspects contribute to good reranking performance and which do not?

3.2 Dataset

We use a dataset provided by ScienceDirect,[3] a popular academic search engine that offers access to millions of academic papers. Users can gain access either by a subscription service, or by individual purchases of papers. The dataset contains anonymized user activity logs from signed in users. We look at newly signed up users and their interactions on the first paper recommendation email. The paper recommendations emails were sent between December 12, 2017 and January 21, 2018. For each user, browsed papers on ScienceDirect prior to receiving the email were also obtained. A browsing action is characterized by any form of a click on a paper, such as a click on the search engine result page, or a click on related papers shown on the detailed paper content page. For email recommendation data, each email contains five candidate paper recommendations where users' responses to each one of them are logged (clicked or not clicked). To obtain paper metadata, we use the paper metadata from Scopus,[4] which can be obtained by querying paper IDs from papers in the ScienceDirect database.

Since we want to study how *content* contributes to better reranking, we need users that have at least a few papers in their browsing history to utilize the content information. The content data of the browsed papers should be clean and complete. Also, we need users who have at least one click on the recommendation email so that we can perform offline evaluations for reranking and calculate the metrics. Correspondingly, we apply the following filtering steps prior to obtaining the data:

(1) we filter out cold start users with fewer than five browsed papers prior to the recommendation;
(2) we remove users whose browsed papers have incomplete or corrupt fields of data; and
(3) we remove the recommendation emails without any clicks.

In total, we have obtained 4,392 recommendation sessions for our experiments. Each session contains one recommendation email with a field that indicates whether each paper has a click, and also the user's browsing history prior to the recommendation's timestamp.

Also readily available are the item-item collaborative filtering scores based on readership of papers from ScienceDirect users. The scores of paper pairs are symmetrical so that $s_{ij} = s_{ji}$.

3.3 Experimental Setup

The experiments on our dataset are conducted through fivefold cross validation. For each run, 4 folds are used for training and 1 fold is used for testing. There are one or more clicks on the candidate paper recommendations for each email. We code relevance as a binary label, which is 1 for clicked papers and 0 for the rest. We compute the mean average precision (MAP) and Precision@k ($k = 1, 2, 3$, denoted as Prec@k for short) as the evaluation metrics.

Significance tests are applied when comparing the results of different models. Specifically, we apply the two-tailed student t test to MAP and Wilcoxon signed rank test to Prec@k, according to assumptions underlying the significance tests.

[3] https://www.sciencedirect.com/.
[4] https://www.scopus.com.

We select the optimal hyper-parameters for HRM by iterating over possible parameter combinations. For the content component of HRM, we have $\theta = 3$ chosen from $\{1, 2, 3, 4, 5\}$; for the behavior component of HRM, we have $\alpha = \lambda = 0.01$ and $k = 100$; for the scoring function of HRM, the hidden layer contains 32 nodes (more nodes may lead to overfitting and worse performances in the experiments), and the learning rate is set as 0.001.

What are appropriate baselines to consider? The first and obvious baseline is the production system that we seek to improve over; this baseline mainly uses item-item similarity-based user browsing data on ScienceDirect. In addition, two families of approaches appear to be natural candidates: *learning to rerank methods* and *collaborative filtering methods*.

As to learning to rerank models, to the best of our knowledge, approaches to learning to rerank a production system published in the literature focus on learning from interaction data (see Section 5.5). We, however, focus on similarity-based models. Thus, we consider an (offline) pointwise learning to rerank model based on logistic regression with Adagrad optimization [15], which has achieved state-of-the art performance [63]. We also use the state-of-the-art pairwise model RankSVM [26] and listwise model LambdaMART [4]. The hyperparameter c_{rank} of RankSVM is selected from $\{0.0001, 0.001, 0.01, 0.1, 1, 10, 100, 1000, 10000\}$. We use the default parameters of LambdaMART in the Ranklib [12] implementation and tune the trees and leaves parameters. We also consider an (offline) linear pairwise learning model that is trained using pairwise hinge loss, which differs from HRM by using a linear scoring function instead of the neural structure. These baselines use the same inputs as HRM.

As to collaborative filtering methods as possible baselines against which to compare the approaches in this article, we compare with libFM [53] and SVDFeature [6]. libFM and SVDFeature construct the feature matrix from user ratings; both can provide effective recommendations even if the ratings are sparse [22, 49].[5]

We describe how to construct the feature matrix for libFM and SVDFeature. (1) The first m values represent the users; (2) the following n values represent the candidate paper recommendations for the user; (3) the next n values represent the browsed papers on the search engine; (4) the final value indicates whether the user clicked the paper from the candidate recommendations. An example is given as follows. Suppose we have 3 users and 10 papers. Suppose for user 1, papers 1, 3, 4, 6, 9 are presented to them as candidate recommendations, among which they clicked paper 1, 3. Besides, the user also browsed papers 2, 5 on the search engine. We use red, blue, green, and black color to represent the users, recommended papers, browsed papers and clicks, respectively, in the feature matrix constructed for user 1, as follows:

$$
\begin{array}{cccccccccccccccccccccc}
1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.7 & 0 & 0 & 0.7 & 0 & 0 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.7 & 0 & 0 & 0.7 & 0 & 0 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.7 & 0 & 0 & 0.7 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0.7 & 0 & 0 & 0.7 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0.7 & 0 & 0 & 0.7 & 0 & 0 & 0 & 0 & 0 & 0 \\
\underbrace{} & & & \underbrace{} & & & & & & & & & \underbrace{} & & & & & & & & & & \\
\text{users} & & & \text{recommended papers} & & & & & & & & & \text{browsed papers} & & & & & & & & & & \text{clicks}
\end{array}
\tag{29}
$$

For SVDFeature, we tune the regularization parameters for user factor λ_1 and item factor λ_2. We tune parameter for user bias b_1 and item bias b_2. We also tune the dimension of latent factors k. The parameters $\lambda_1, \lambda_2, b_1, b_2$ are explored from $\{0.01, 0.1, 1, 10\}$; k is explored from $\{5, 10, 15, 20\}$.

[5]The rating density is less than 0.02% even if we consider both browses and clicks as ratings on papers, which is significantly less than common recommendation datasets (Movielens 100K: 6.30%, Movielens 1M: 4.47%, FilmTrust: 1.14%).

Table 2. Results of Reranking Candidate Paper Recommendations Across Models

Model	MAP	Prec@1	Prec@2	Prec@3	W/T/L
Production baseline	0.588	0.392	0.350	0.323	-/-/-
libFM	0.525	0.302	0.295	0.293	1,854/999/1,539
SVDFeature	0.525	0.305	0.296	0.292	1,837/1,004/1,551
Linear pointwise learning to rerank	0.534	0.330	0.296	0.280	1,595/753/2,044
Linear pairwise learning to rerank	0.620	0.432	0.378	0.343	1,822/1,254/1,316
RankSVM	0.615	0.423	0.376	0.341	1,924/1,220/1,248
LambdaMART	0.627	0.443	0.383	0.345	2,006/1,102/1,284
HRM	0.663	0.502	0.453	0.421	2,005/1,171/1,216

Win/**T**ie/**L**oss are the number of users for which a model performs better than, the same as, or worse than the production baseline.

We use Stochastic Gradient Descent to optimize libFM. We tune the regularization parameters for bias α, one-way interaction β, two-way interaction λ, and the dimension of latent factors k. Similarly, α, β, λ are tuned from $\{0.01, 0.1, 1, 10\}$ and k from $\{5, 10, 15, 20\}$.

Note that when answering the second and third research questions (examining the feature utility), certain components' feature size is very small. For instance, the behavior component consists of a feature size of two: one from our proposed behavioral model (Section 2.2.5), one from the browsing similarity (Section 2.1). Such a small feature vector is not suitable as input for the neural structure in HRM. Therefore, we use the pairwise linear model in this case.

4 RESULTS AND ANALYSES

In this section, we present the experimental results, including the results of different models, and break-down analyses on different components of the model.

4.1 Overall Comparison

To address our first research question (Does HRM, which utilizes content and behavior, provide improvements in reranking over the production baseline?), we compare HRM against the production baseline, as well as the other baselines, see Table 2.

Compared with all baselines, significant improvements are made in the hybrid reranking model HRM that combines content and behavior ($p < 0.01$). Compared with the production baseline, HRM performs better or the same for 72.3% of the users. There is a relative 13% increase in MAP and a relative 28% increase in Prec@1 for HRM, meaning that users are more likely to click the top candidates in the reranked list. This answers the first research question.

Besides, when given the same input features, HRM also performs better than all four reranking baselines, although leading LambdaMART only by a small margin. LambdaMART is a strong baseline and it has more users that perform better than the production baseline, compared with HRM. Interestingly, the pointwise learning to rerank method is beaten by the production baseline on all metrics. This shows that learning absolute user preferences of papers based on clicks is not optimal in our scenario. Models based on pairwise and listiwise learning (HRM, LambdaMART, RankSVM, and the linear pairwise model) have produced better results by learning relative user preferences.

The behavioral baselines, i.e., libFM and SVDFeature demonstrate worse performance than HRM. A possible explanation is that libFM and SVDFeature cannot utilize paper browsing similarity, which contains useful information to recover user behavior patterns, and they also do not capture content similarities.

Table 3. The Performance of Reranking Candidate Paper Recommendations Using Different Input Features of HRM Shown in Figure 2

Model	MAP	Prec@1	Prec@2	Prec@3
proposed behavior feature	0.540	0.332	0.302	0.288
all behavior	0.602	0.411	0.358	0.327
only recent content	0.590	0.384	0.354	0.333
only historical content	0.582	0.374	0.343	0.327
all content without attention	0.598	0.398	0.359	0.337
all content	0.601	0.402	0.365	0.338

We have also attempted to replace our behavior model in HRM by libFM and SVDFeature to see how they work with content similarities, which yields worse results than the original HRM. The scores are neglected for brevity.

4.2 Utility of Content and Behavior Features in Reranking

To answer the second research question (What is the utility of the content features and behavior features in reranking, respectively?), we analyze the utility of the input features of individual components in HRM, shown in Figure 2. Specifically, we look at the reranking performance using the following input features separately.

- Proposed behavior feature (Section 2.2.5).
- All behavior features: browsing similarity features (Section 2.1) and proposed behavior feature (Section 2.2.5).
- Only recent content similarity: S_{recent} (Section 2.2.4).
- Only historical content similarity: $S_{history}$ (Section 2.2.4).
- All content features without attention features: S_{recent}, $S_{history}$ (Section 2.2.4).
- All content features: S_{recent}, $S_{history}$, attention features (Section 2.2.4).

The behavior features have small sizes (a single feature from our behavioral model and the production system, respectively). Therefore, we opt for the linear pairwise model, because the small input feature vector is not suitable for the neural structure in HRM. For other features that have larger sizes, we use the neural structure of HRM for reranking; see Table 3 for the results.

Using behavior and content separately for reranking, the results (MAP score of 0.602 and 0.601, respectively) already outperform the production baseline (0.588) that mainly uses item-item collaborative filtering. The proposed behavior feature provides a boost for the behavior component in addition to using the browsing similarity features from the production system ($p < 0.01$). However, the content component has a performance quite close to the behavior component. The attention features lead to a slight improvement over the model without them. We also find that using the recently browsed papers is better for reranking paper candidates than to using historically browsed papers, and even better is to use both recently and historically browsed papers. This answers the second research question.

4.3 Utility of Paper Aspects in Reranking

To answer the third research question (Within the content features, for paper similarity based on various paper aspects, which paper aspects contribute to good reranking performance and which do not?), we continue to delve into the content similarity in HRM, which contains similarity measures for different aspects of papers. We are interested to see the reranking performance of features based on a single paper aspect. For each paper aspect, we take the recent/historic similarity and

Table 4. Reranking Candidate Paper Recommendations by Restricting the Pairwise Linear Learning Rerank Model to Using Only One Paper Aspect

Field	MAP	Prec@1	Prec@2	Prec@3
freshness	0.426	0.153	0.248	0.242
popularity	0.453	0.154	0.276	0.284
venue	0.468	0.203	0.272	0.276
impact	0.489	0.257	0.283	0.267
word	0.526	0.312	0.291	0.284
author	0.549	0.327	0.320	0.311
paper entity	0.550	0.330	0.319	0.311

the recent/historical attention scores as the input features for reranking. Similar to Section 4.2, we use the pairwise linear model due to the small input feature size. The results are shown in Table 4.

The reranking performance of the paper candidates differs among the paper aspects. In general, the similarity measures based on semantics or entities perform better than those that do not. The two entity space measures: the author and paper entity similarities perform better than other measures, also beating the word-space similarity. Comparing three entity-based measures, the author similarity performs similarly to the paper entity similarity, this is due to the high correlation between them (Pearson correlation coefficient being 0.88); the author similarity performs much better than the venue similarity (0.549 vs. 0.468 for MAP scores). This may suggest that users pay attention to the authorship of the paper more than the venue. Using freshness, popularity, or impact similarity alone does not generate good performance, understandably, as these measures do not consider semantic relevance or entity relationships. Combining all paper aspects produces the best performance. The third research question is hence answered by the above comparisons of paper aspects' utility in reranking.

5 RELATED WORK

In this section, we discuss the related work to our study. The related work spans several topics: academic search, paper recommendation, citation recommendation, top-N recommendation, and learning to rerank the output of a production system. We introduce them below and explain how they are related to our work.

5.1 Academic Search

Our work is relevant to academic search, because we are examining the recommendation service attached to an academic search engine. Academic search engines [20, 35, 59, 66] have given users convenient access to academic resources such as papers, journals, and authors. Mitra and Awekar [44] found that different academic search engines have their own coverage of literature and ranking strategy, and the overlap among search results is low. Compared to general web search, there is far less research on user behavior in academic search, possibly due to a lack of public datasets. Research on academic search has examined user behavior through surveys [48, 51, 52] and aggregated usage statistics such as query frequencies [29]. Khabsa et al. [30] studied user queries on Microsoft Academic Search and proposed a query classifier.

Recently, more studies have been conducted on user behavior within and across search sessions, based on a large-scale user transaction log. Li et al. [39] have studied the null query phenomenon in academic search and proposed a query suggestion method as a remedy. Li and de Rijke [37]

have revealed correlations of query reformulation and topic shift in academic search. Li and de Rijke [36] have studied characteristics of user queries in academic search following major scientific events. Li and de Rijke [38] have also studied download behavior in academic search and proposed a download prediction model. There has also been research aimed to improve the search experience. Tang et al. [67] combined topic modeling with random walks to improve academic search retrieval performance. Khazaei and Hoeber [31] proposed a visual search interface via citation links to help users better navigate through search results. Xiong et al. [72] proposed improving paper rankings in academic search using entity embeddings.

Our work in this article differs from previous work in academic search in that we do not directly deal with search. We utilize the browsing history on the academic search engine to make improvements to a paper recommender.

5.2 Academic Paper Recommendation

Our recommendation task falls in the broad category of academic paper recommendations. Generally, based on the system inputs, paper recommendation tasks can be classified into the following scenarios: the system generates a list of paper recommendations given a single paper as input [2, 25, 46]; the system generates a list of paper recommendations given a set of papers as input (without ordering) [33, 62, 68]; the system generates recommendations given a time-ordered set of papers as input [23, 71]. The first and second scenarios include cases where a user is browsing a paper, or a list of relevant papers is available (e.g., through a set of papers selected by the user). The system assumes the input to be representative of a user's interests, then provides related papers as recommendations. These are the most common scenarios that are being studied. The third scenario is rarely studied, because: (1) it is relatively difficult to acquire user data that spans a long period, for instance, users' paper browsing history; (2) it is more difficult to model user interests based on a sequence of inputs, compared to static inputs in the first two scenarios.

Common methods involved in making recommendations can be classified as: content-based filtering (CBF), collaborative filtering (CF) and hybrid models that combine the two. CBF involves using various parts of the paper contents, such as titles, abstracts, and keywords, to suggest related papers based on their similarity with input paper(s) [19, 25, 65]. While they are able to expose related papers that are similar by content, CBF models do not take into account user-paper interactions. CF models, however, utilize the user-paper interactions to generate recommendations, and can result in strong performance [5, 23, 50]. However, a common drawback of CF models is the cold start problem, which is severe in our academic recommendations when using real user-paper interaction data. Finally, there are hybrid models that combine CBF and CF models for paper recommendations [17, 68, 69]. The hybridization process is usually rule-based instead of learned: either the system first runs CBF models and then uses its output as input to run CF models to generate recommendations (cascade hybrid); or it simply mixes results that are separately generated from CBF and CF models (mixed hybrid).

Our work in this article differs from previous work on academic paper recommendation in that we study a rarely examined, but real scenario: generating paper recommendations given an ordered sequence as input. Specifically, we make recommendations for new users that sign up for the recommendations based on their browse history on the search engine. Compared to Reference [71], which uses a simulated and artificial recommendation setting, our scenario concerns real user interactions with a recommender system. We have proposed a hybrid model that combines content similarities, that draws distinction between multiple aspects of paper contents, and behavior-based similarities. We have applied pointwise and pairwise learning approach to train the model, unlike the rule-based approaches to generate paper recommendation that do not apply learning techniques [17, 68].

5.3 Citation Recommendation

Citation recommendation is sometimes mixed with paper recommendation. Hence, we draw the distinction between our paper recommendation task and citation recommendation. We consider citation recommendation to be the task of recommending papers to an author who is writing a manuscript. A citation recommender may take a complete or incomplete manuscript as input, identify places where citations are needed, and recommend relevant citations [21, 64]. It may also take a piece of "context" as input, which is represented as a few sentences, and generate relevant citation suggestions [16, 24].

It is obvious that the citation recommendation task is mainly focused on similarity. Even when collaborative filtering is applied, it is using the citation relation matrix as a paper similarity measure [42], instead of using the user-paper rating matrix. The evaluation setup is also confined to predicting the cited papers of an input paper or paragraph.

Our work in this article differs from previous work on citation recommendations in terms of the methods we propose, the recommendation goal, and the evaluation setup.

5.4 Top-*N* Recommendation

In the context of more general recommendation problems, our scenario is related to top-*N* recommendation [14]. Top-*N* recommender systems provide users with a ranked list of items based on predicted scores of individual items, where the relative ranking matters more than the absolute item scores. This is similar to our problem, as we aim to produce a ranking of papers according to the predicted scores. However, the candidate set from which we make recommendations is different: We pick the papers from a recommendation email, while a typical top-*N* recommender selects from all items that have not been rated by users.

Top-*N* recommenders have been intensively studied [55]. In general, there are approaches that use latent space models [11] and approaches that rely on neighborhood-based models (whether user-based or item-based) [14]. While latent factor models can also generate top-*N* recommendations, they are originally designed for the rating prediction task. Therefore, they are sub-optimal for top-*N* recommendation. Neighborhood-based methods identify similar users or items, and have been shown to be more suitable for the top-*N* recommendation problem [1, 14, 27, 47]. Item-based methods have been shown to outperform user-based methods for the top-*N* recommendation task [8]. Similarity models have recently been proposed to improve item-based neighborhood models. They learn a coefficient matrix that is analogous to the item-item similarities [7, 27, 28, 47] directly from the data. A novel similarity model, Sparse Linear Method (SLIM), has been proposed by Reference [47]. Several authors have proposed improvements to SLIM. Low-rankness has been investigated to capture transitive relations [7, 27, 28]. Kabbur et al. [27] proposed the Factored Item Similarity Model (FISM), which factorizes the coefficient matrix into two low-dimensional factor matrices. Cheng et al. [7] proposed the Low-rank Sparse Linear Method (LorSLIM), which introduces a rank regularization to SLIM. Kang and Cheng [28] made improvements over LorSLIM by providing a better proxy to approximate the rank of the coefficient matrix. Instead of estimating a single model for all users, Christakopoulou and Karypis [8] clustered users and estimated several local models. Zhao and Guo [73] minimized a combined heterogeneous loss function, which is a combination of pair-wise ranking loss and point-wise recovery loss. Wu et al. [70] generalized FISM from linear to non-linear by incorporating a denoising auto-encoder.

Our work in this article differs from previous work on top-*N* recommendation in important ways. First, directly applying top-*N* recommendation models to our task will lead to two problems: new users have no clicks on the recommendation emails, a situation that cannot be handled by existing top-*N* recommenders. Also, we have two types of interaction between users and

items: user browses and user clicks. Existing top-N recommenders focus only on homogeneous interactions.

5.5 Reranking the Output of a Production System

Like us, Lefortier et al. [34], Moon et al. [45], and Zoghi et al. [74] use a commercial search engine as their main baseline that they learn to improve. Moon et al. [45] and Lefortier et al. [34]'s methods directly use click-through rates, with a focus on documents that appear in the first position; both also focus on recency ranking and queries with shifting intent. Zoghi et al. [74] learn from a pairwise signal – out of order clicks in the top 5 produced by the production ranker.

Our work in this article differs from previous work on reranking the output of a production search engine or recommender system in that we do not restrict ourselves to recency ranking. Moreover, we do not work in an online setting and we do include content-based signals, not just behavior-based ones.

6 CONCLUSION AND DISCUSSION

In this article, we have examined an interesting recommendation scenario for an academic search engine, namely, to rerank paper recommendations in email newsletters for newly signed up users. We have addressed this challenge by proposing a hybrid recommendation approach that includes a content component and a behavior component. The content component measures similarities of various paper aspects between users' browsed articles and candidate recommendations, and also considers the user's attention on paper aspects and on recent/historical browsing. The behavior component learns a mapping from browsed articles to user clicks in the recommendations. The model combines content and behavior through a pairwise learning approach that is based on user interaction data.

We have found that our hybrid reranking model HRM significantly improves over the production baseline. We have dug into the components of our model to see what works and what does not. In the content component, the graph embeddings work the best, especially the author similarity based on soft matching; users' recently browsed articles can lead to better recommendations compared to historical browsing; however, popularity and impact similarities are not sufficient to bring up good recommendations alone. In the behavior component, our learned scores combined with browsing similarity scores have led to better performance than the production baseline. The best performance is achieved when combining content and behavior through learning.

Our hybrid reranking model HRM can be seen as a module that can be plugged into a recommendation system. Besides, we also have generalizable insights for other paper recommenders. For instance, we have revealed how each paper aspect contributes to the reranking performance.

A limitation of our study is that we have not performed online evaluations, such as A/B testing, to validate the model's effect on user engagement. Another limitation is due to the production dataset: our reranking is limited to the candidate articles generated by the production system. Therefore, if the inputs are not of high quality, it will impact our final recommendation performance. In practice, HRM could be used to rerank a longer list of candidate recommendations, so that it effectively chooses the top five to be sent in an email. It would be interesting to also explore different methods of paper candidate generation and examine how they impact the recommendation performance. Besides, if we can obtain user profile information (such as domain interests), can we apply topic modeling to provide more personalized recommendations for users? We leave these interesting questions as future work.

ACKNOWLEDGMENTS

We thank our anonymous reviewers for their constructive comments and helpful suggestions.

REFERENCES

[1] Fabio Aiolli. 2013. A preliminary study on a recommender system for the million songs dataset challenge. In *Proceedings of the 4th Italian Information Retrieval Workshop (CEUR Workshop Proceedings)*, Vol. 964. CEUR-WS.org, 73–83. Retrieved from http://ceur-ws.org/Vol-964/paper12.pdf.

[2] Joeran Beel, Akiko Aizawa, Corinna Breitinger, and Bela Gipp. 2017. Mr. DLib: Recommendations-as-a-Service (RaaS) for academia. In *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries (JCDL'17)*. IEEE, 1–2.

[3] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Adv. Neural Info. Process. Syst.* 2787–2795.

[4] Christopher J. C. Burges. 2010. From RankNet to LambdaRank to LambdaMart: An overview. *Learning* 11, 23–581 (2010), 81.

[5] Laurent Charlin, Richard S. Zemel, and Hugo Larochelle. 2014. Leveraging user libraries to bootstrap collaborative filtering. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 173–182.

[6] Tianqi Chen, Weinan Zhang, Qiuxia Lu, Kailong Chen, Zhao Zheng, and Yong Yu. 2012. SVDFeature: A toolkit for feature-based collaborative filtering. *J. Mach. Learn. Res.* 13 (2012), 3619–3622.

[7] Yao Cheng, Li'ang Yin, and Yong Yu. 2014. LorSLIM: Low rank sparse linear methods for top-N recommendations. In *Proceedings of the 2014 IEEE International Conference on Data Mining (ICDM'14)*. IEEE Computer Society, 90–99.

[8] Evangelia Christakopoulou and George Karypis. 2016. Local item-item models for top-N recommendation. In *Proceedings of the 10th ACM Conference on Recommender Systems*. ACM, 67–74.

[9] Fan R. K. Chung. 1997. *Spectral Graph Theory*. Number 92. American Mathematical Soc.

[10] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep neural networks for YouTube recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*. ACM, 191–198.

[11] Paolo Cremonesi, Yehuda Koren, and Roberto Turrin. 2010. Performance of recommender algorithms on top-n recommendation tasks. In *Proceedings of the 2010 ACM Conference on Recommender Systems, RecSys 2010*. ACM, 39–46.

[12] Van Dang. 2018. The Lemur Project-Wiki-RankLib. Lemur Project. Retrieved from https://sourceforge.net/projects/lemur/.

[13] James Davidson, Benjamin Liebald, Junning Liu, Palash Nandy, Taylor Van Vleet, Ullas Gargi, Sujoy Gupta, Yu He, Mike Lambert, Blake Livingston, and Dasarathi Sampath. 2010. The YouTube video recommendation system. In *Proceedings of the 4th ACM Conference on Recommender Systems*. ACM, 293–296.

[14] Mukund Deshpande and George Karypis. 2004. Item-based top-*N* recommendation algorithms. *ACM Trans. Info. Syst.* 22, 1 (2004), 143–177.

[15] John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.* 12 (2011), 2121–2159.

[16] Travis Ebesu and Yi Fang. 2017. Neural citation network for context-aware citation recommendation. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 1093–1096.

[17] Michael D. Ekstrand, Praveen Kannan, James A. Stemper, John T. Butler, Joseph A. Konstan, and John T. Riedl. 2010. Automatically building research reading lists. In *Proceedings of the 4th ACM Conference on Recommender Systems*. ACM, 159–166.

[18] Asmaa Elbadrawy and George Karypis. 2015. User-specific feature-based similarity models for top-*n* recommendation of new items. *ACM Trans. Intell. Syst. Technol.* 6, 3 (2015), 33:1–33:20.

[19] Felice Ferrara, Nirmala Pudota, and Carlo Tasso. 2011. A keyphrase-based paper recommender system. In *Italian Research Conference on Digital Libraries*. Springer, 14–25.

[20] Google Scholar. 2018. Retrieved from https://scholar.google.com/.

[21] Qi He, Daniel Kifer, Jian Pei, Prasenjit Mitra, and C. Lee Giles. 2011. Citation recommendation without author supervision. In *Proceedings of the 4th ACM International Conference on Web Search and Data Mining*. ACM, 755–764.

[22] Xiangnan He and Tat-Seng Chua. 2017. Neural factorization machines for sparse predictive analytics. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 355–364.

[23] Maya Hristakeva, Daniel Kershaw, Marco Rossetti, Petr Knoth, Benjamin Pettit, Saúl Vargas, and Kris Jack. 2017. Building recommender systems for scholarly information. In *Proceedings of the 1st Workshop on Scholarly Web Mining*. ACM, 25–32.

[24] Wenyi Huang, Zhaohui Wu, Liang Chen, Prasenjit Mitra, and C. Lee Giles. 2015. A neural probabilistic model for context-based citation recommendation. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence*. AAAI, 2404–2410.

[25] Yichen Jiang, Aixia Jia, Yansong Feng, and Dongyan Zhao. 2012. Recommending academic papers via users' reading purposes. In *Proceedings of the 6th ACM Conference on Recommender Systems*. ACM, 241–244.

[26] Thorsten Joachims. 2006. Training linear SVMs in linear time. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 217–226.

[27] Santosh Kabbur, Xia Ning, and George Karypis. 2013. FISM: Factored item similarity models for top-N recommender systems. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 659–667.

[28] Zhao Kang and Qiang Cheng. 2016. Top-N recommendation with novel rank approximation. In *Proceedings of the 2016 SIAM International Conference on Data Mining*. SIAM, 126–134.

[29] Hao-Ren Ke, Rolf Kwakkelaar, Yu-Min Tai, and Li-Chun Chen. 2002. Exploring behavior of e-journal users in science and technology: Transaction log analysis of elsevier's sciencedirect OnSite in Taiwan. *Library Info. Sci. Res.* 24, 3 (2002), 265–291.

[30] Madian Khabsa, Zhaohui Wu, and C. Lee Giles. 2016. Towards better understanding of academic search. In *Proceedings of the 16th ACM/IEEE-CS on Joint Conference on Digital Libraries*. ACM, 111–114.

[31] Taraneh Khazaei and Orland Hoeber. 2017. Supporting academic search tasks through citation visualization and exploration. *Int. J. Dig. Libraries* 18, 1 (2017), 59–72.

[32] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[33] Onur Küçüktunç, Erik Saule, Kamer Kaya, and Ümit V. Çatalyürek. 2012. Recommendation on academic networks using direction aware citation analysis. *arXiv preprint arXiv:1205.1143* (2012).

[34] Damien Lefortier, Pavel Serdyukov, and Maarten de Rijke. 2014. Online exploration for detecting shifts in fresh intent. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*. ACM, 589–598.

[35] Huajing Li, Isaac Councill, Wang-Chien Lee, and C. Lee Giles. 2006. CiteSeerx: An architecture and web service design for an academic document search engine. In *Proceedings of the 15th International Conference on World Wide Web*. ACM, 883–884.

[36] Xinyi Li and Maarten de Rijke. 2017. Academic search in response to major scientific events. In *Proceedings of the 5th International Workshop on Bibliometric-enhanced Information Retrieval*.

[37] Xinyi Li and Maarten de Rijke. 2017. Do topic shift and query reformulation patterns correlate in academic search? In *Proceedings of the 39th European Conference on IR Research*. Springer, 146–159.

[38] Xinyi Li and Maarten de Rijke. 2019. Characterizing and predicting downloads in academic search. *Info. Process. Manage.* 56, 3 (2019), 394–407.

[39] Xinyi Li, Bob J. A. Schijvenaars, and Maarten de Rijke. 2017. Investigating queries and search failures in academic search. *Info. Process. Manage.* 53, 3 (May 2017), 666–683.

[40] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, Vol. 15. 2181–2187.

[41] Greg Linden, Brent Smith, and Jeremy York. 2003. Amazon.Com recommendations: Item-to-item collaborative filtering. *IEEE Internet Comput.* 7, 1 (Jan. 2003), 76–80.

[42] Haifeng Liu, Xiangjie Kong, Xiaomei Bai, Wei Wang, Teshome Megersa Bekele, and Feng Xia. 2015. Context-based collaborative filtering for citation recommendation. *IEEE Access* 3 (2015), 1695–1703.

[43] Hao Ma, Dengyong Zhou, Chao Liu, Michael R. Lyu, and Irwin King. 2011. Recommender systems with social regularization. In *Proceedings of the 4th International Conference on Web Search and Web Data Mining*. ACM, 287–296.

[44] Anasua Mitra and Amit Awekar. 2017. On low overlap among search results of academic search engines. In *Proceedings of the 26th International Conference on World Wide Web Companion*. ACM, 823–824.

[45] Taesup Moon, Wei Chu, Lihong Li, Zhaohui Zheng, and Yi Chang. 2012. An online learning framework for refining recency search results with user click feedback. *ACM Trans. Info. Syst.* 30, 4 (2012), 20:1–20:28.

[46] Cristiano Nascimento, Alberto H. F. Laender, Altigran S. da Silva, and Marcos André Gonçalves. 2011. A source independent framework for research paper recommendation. In *Proceedings of the 11th Annual International ACM/IEEE Joint Conference on Digital Libraries*. ACM, 297–306.

[47] Xia Ning and George Karypis. 2011. SLIM: Sparse linear methods for top-N recommender systems. In *Proceedings of the 11th IEEE International Conference on Data Mining*. IEEE Computer Society, 497–506.

[48] Xi Niu and Bradley M. Hemminger. 2012. A study of factors that affect the information-seeking behavior of academic scientists. *J. Amer. Soc. Info. Sci. Technol.* 63, 2 (2012), 336–353.

[49] Zhen Pan, Enhong Chen, Qi Liu, Tong Xu, Haiping Ma, and Hongjie Lin. 2016. Sparse factorization machines for click-through rate prediction. In *Proceedings of the 16th International Conference on Data Mining*. IEEE Computer Society, 400–409.

[50] David M. Pennock, Eric Horvitz, Steve Lawrence, and C. Lee Giles. 2000. Collaborative filtering by personality diagnosis: A hybrid memory-and model-based approach. In *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*. Morgan Kaufmann Publishers Inc., 473–480.

[51] Sheila Pontis and Ann Blandford. 2015. Understanding "Influence": An exploratory study of academics' processes of knowledge construction through iterative and interactive information seeking. *J. Assoc. Info. Sci. Technol.* 66, 8 (2015), 1576–1593.

[52] Sheila Pontis, Ann Blandford, Elke Greifeneder, Hesham Attalla, and David Neal. 2015. Keeping up to date: An academic researcher's information journey. *J. Amer. Soc. Info. Sci. Technol.* 68, 1 (2015), 22–35.
[53] Steffen Rendle. 2012. Factorization machines with libFM. *ACM Trans. Intell. Syst. Technol.* 3, 3 (2012), 57:1–57:22.
[54] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence*. AUAI Press, 452–461.
[55] Francesco Ricci, Lior Rokach, and Bracha Shapira (Eds.). 2015. *Recommender Systems Handbook*. Springer.
[56] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2000. Analysis of recommendation algorithms for e-commerce. In *Proceedings of the 2nd ACM Conference on Electronic Commerce*. ACM, 158–167.
[57] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th International Conference on World Wide Web*. ACM, 285–295.
[58] Martin Saveski and Amin Mantrach. 2014. Item cold-start recommendations: Learning local collective embeddings. In *Proceedings of the 8th ACM Conference on Recommender Systems*. ACM, 89–96.
[59] ScienceDirect. 2015. Retrieved from https://sciencedirect.com.
[60] ScienceDirect. 2016. Retrieved from https://www.elsevier.com/solutions/sciencedirect/features.
[61] Semantic Scholar. 2018. Retrieved from https://www.semanticscholar.org/.
[62] Aravind Sesagiri Raamkumar, Schubert Foo, and Natalie Pang. 2018. Can I have more of these please? assisting researchers in finding similar research papers from a seed basket of papers. Emerald Publishing Limited.
[63] Guocong Song. 2014. Point-wise approach for yandex personalized web search challenge. In *Proceedings of the WSDM 2014 Workshop on Web Search Click Data*. ACM.
[64] Trevor Strohman, W. Bruce Croft, and David Jensen. 2007. Recommending citations for academic papers. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 705–706.
[65] Kazunari Sugiyama and Min-Yen Kan. 2010. Scholarly paper recommendation via user's recent research interests. In *Proceedings of the 10th Annual Joint Conference on Digital Libraries*. ACM, 29–38.
[66] Jie Tang. 2016. AMiner: Toward understanding big scholar data. In *Proceedings of the 9th ACM International Conference on Web Search and Data Mining*. ACM, 467–467.
[67] Jie Tang, Ruoming Jin, and Jing Zhang. 2008. A topic modeling approach and its integration into the random walk framework for academic search. In *Proceedings of the 8th IEEE International Conference on Data Mining*. IEEE, 1055–1060.
[68] Roberto Torres, Sean M. McNee, Mara Abel, Joseph A. Konstan, and John Riedl. 2004. Enhancing digital libraries with TechLens+. In *Proceedings of the 4th ACM/IEEE-CS Joint Conference on Digital Libraries*. ACM, 228–236.
[69] Chong Wang and David M. Blei. 2011. Collaborative topic modeling for recommending scientific articles. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 448–456.
[70] Yao Wu, Christopher DuBois, Alice X. Zheng, and Martin Ester. 2016. Collaborative denoising auto-encoders for top-N recommender systems. In *Proceedings of the 9th ACM International Conference on Web Search and Data Mining*. ACM, 153–162.
[71] Zhibo Xiao, Feng Che, Enuo Miao, and Mingyu Lu. 2014. Increasing serendipity of recommender system with ranking topic model. *Appl. Math. Info. Sci.* 8, 4 (2014), 2041.
[72] Chenyan Xiong, Russell Power, and Jamie Callan. 2017. Explicit semantic ranking for academic search via knowledge graph embedding. In *Proceedings of the 26th International Conference on World Wide Web*. ACM, 1271–1279.
[73] Feipeng Zhao and Yuhong Guo. 2016. Improving top-N recommendation with heterogeneous loss. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence*. IJCAI/AAAI Press, 2378–2384.
[74] Masrour Zoghi, Tomáš Tunys, Lihong Li, Damien Jose, Junyan Chen, Chun Ming Chin, and Maarten de Rijke. 2016. Click-based hot fixes for underperforming torso queries. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 195–204.

Received July 2018; revised February 2019; accepted February 2019