Learning a Joint Search and Recommendation Model from User-Item Interactions

Hamed Zamani*
Center for Intelligent Information Retrieval
University of Massachusetts Amherst
Amherst, MA 01003
zamani@cs.umass.edu

W. Bruce Croft
Center for Intelligent Information Retrieval
University of Massachusetts Amherst
Amherst, MA 01003
croft@cs.umass.edu

ABSTRACT

Existing learning to rank models for information retrieval are trained based on explicit or implicit query-document relevance information. In this paper, we study the task of learning a retrieval model based on user-item interactions. Our model has potential applications to the systems with rich user-item interaction data, such as browsing and recommendation, in which having an accurate search engine is desired. This includes media streaming services and e-commerce websites among others. Inspired by the neural approaches to collaborative filtering and the language modeling approaches to information retrieval, our model is jointly optimized to predict user-item interactions and reconstruct the item textual descriptions. In more details, our model learns user and item representations such that they can accurately predict future user-item interactions, while generating an effective unigram language model for each item. Our experiments on four diverse datasets in the context of movie and product search and recommendation demonstrate that our model substantially outperforms competitive retrieval baselines, in addition to providing comparable performance to state-of-the-art hybrid recommendation models.

ACM Reference Format:
Hamed Zamani and W. Bruce Croft. 2020. Learning a Joint Search and Recommendation Model from User-Item Interactions. In *The Thirteenth ACM International Conference on Web Search and Data Mining (WSDM '20), February 3–7, 2020, Houston, TX, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3336191.3371818

1 INTRODUCTION

Learning to rank models have been successfully employed for various retrieval tasks, such as web search, personal search, and question answering [25]. They are mostly trained with either explicit query-document relevance signals, or implicit feedback collected from the user interactions with the retrieval system. Existing learning to rank models heavily rely on query-document (or user-query-document) interactions, such as clickthrough data. However, there exist other types of user interactions with the system that can be potentially useful in developing retrieval models. For instance, in many scenarios, users have countless interactions with the items without using the search engine, e.g., browsing and clicking on items, or interacting with the outputs of recommendation engine.

Although user-item interactions have been utilized for user modeling in recommender systems [17, 38, 43] and search personalization [28, 33, 47], learning a retrieval model from user-item interaction data has not yet been explored. Belkin and Croft [4] pointed out the similarities and unique challenges of information retrieval (IR) and information filtering systems, since 1990s. They concluded that their underlying goals are essentially equivalent, and thus they are two sides of the same coin. This has inspired Zamani and Croft [52] to develop a preliminary model for joint optimization of search and recommendation models that learns from both user-item and query-item interactions. This paper extends their work by learning a joint search and recommendation model from user-item interactions.

Learning an accurate retrieval model from user-item interactions (i.e., collaborative filtering data) has several real-world applications. For example, in media streaming services, such as Netflix and Spotify, where rich user-item interaction data exists, building an accurate search engine learned from such large-scale data is desired.

In this paper, we propose a model that learns to predict user-item interactions (collaborative filtering) and reconstruct the items' description text based on their learned representations. In other words, our model learns item representations that not only simulate user behaviors to predict future user-item interactions, but can also be mapped to a natural language space for further retrieval purposes. The recommendation component of our model is based on neural collaborative filtering [15]. The item reconstruction component is developed based on relevance-based word embedding [51] to estimate a language model for each item. We focus on the unigram language model that has been widely explored in the IR literature [35, 56]. These models can easily be extended to ngram models to capture important phrases [7].

In our experiments, we train our model using four datasets in the context of movie and product search and recommendation, including MovieLens 20M [13] and three diverse categories of Amazon products [16, 29]. To evaluate the retrieval performance of the model, we created a movie retrieval dataset with over 900 real queries and manual relevance annotation for the movies included in the MovieLens 20M dataset. For product retrieval, we followed the evaluation methodology used in prior work [2, 48] using the Amazon products data. Our experiments demonstrate that our model substantially outperforms previous competitive retrieval baselines,

*Hamed Zamani is currently affiliated with Microsoft.

WSDM '20, February 3–7, 2020, Houston, TX, USA

ACM ISBN 978-1-4503-6822-3/20/02…$15.00
https://doi.org/10.1145/3336191.3371818

while performing on par with state-of-the-art hybrid recommendation models. Our work closes the gap between search and recommendation models and allows further investigation of employing IR technologies and recommendation data and the other way around.

In summary, the major contributions of this paper include:

(1) Introducing the notion of learning a retrieval model from user-item interactions, and proposing a joint search and recommendation model that is solely trained based on user-item interactions and item descriptions.
(2) Discussing the connection between the proposed model and matrix factorization.
(3) Developing manual relevance annotations for over 900 real queries collected from Yahoo! Answers under the movies category, in which the relevant items (i.e., correct answers) are manually linked to the MovieLens 20M movie IDs. Our dataset is publicly available for research purposes.
(4) Results showing that our model substantially outperforms the retrieval baselines. The recommendation model also outperforms state-of-the-art collaborative filtering models and performs on par with competitive hybrid recommendation models. Our analysis reveals the reasons for the obtained improvements.
(5) Discussing potential applications of the developed model in multiple important and active research areas.

2 RELATED WORK

User-item interactions have been widely used to train and evaluate recommendation algorithms. Developing effective recommender systems mostly relies on accurate modeling of user preferences on items based on the past interactions, which is called collaborative filtering [17, 38, 43]. Matrix factorization models [19, 22] are the most popular collaborative filtering approaches that learn low-dimensional representations for users and items.

Recently, neural collaborative filtering methods have shown two major advantages over conventional matrix factorization methods. The first one is their flexibility in utilizing different types of information to improve recommendation [57]. For instance, Ai et al. [2] utilized textual information provided by user reviews in a neural recommendation model and showed significant improvements. Their ability to produce explanations has also been studied in [18]. The second advantage is their superior performance. The winning approaches in recent competitions, such as ACM RecSys Challenge 2018 [6], are mainly or partially based on neural network models [55], which have been used earlier for recommendation tasks in [42]. He et al. [15] recently proposed a simple neural collaborative filtering model. They showed that a part of their model is a generalized version of matrix factorization models, which clearly explains their superior performance. Because of these two advantages, neural networks have been recently adapted for different recommendation tasks, such as collaborative [10, 15] and content-based filtering [3].

Hybrid recommendation algorithms use side information (e.g., item description, user-generated tags and reviews, etc.) alongside user-item interaction data to improve the performance of collaborative filtering models, especially for cold-start users. They mostly use side information as the input of their system, which is different from the way that we use item description in our model. The goal in hybrid recommendation is improving the recommendation performance, while our goal is to learn natural language representations for items for retrieval purposes. We use a word embedding component in our model. Word embedding has been previously applied to different recommendation algorithms. For instance, Zagheli et al. [36] proposed a semantic-aware user modeling algorithm for text recommendation that uses pre-trained word embedding models. In contrast to previous work, we use relevance-based word embedding [51] to learn a unigram language model for each item using the user-item interaction data.

User-item interaction has been used for user modeling with the goal of search result personalization. For example, browsing history can be seen as an example of user-item interaction in the context of web search. Both short- and long-term click and browsing history has been used for search personalization [28, 47]. Noll and Meinel [33] used social bookmarking for modeling user profiles for search personalization.

Recently, Zamani and Croft [52] proposed to model search and recommendation models as a joint optimization problem. They presented a set of preliminary results which show that search and recommendation models can both benefit from the data collected for each system. Unlike their work, we do not use any query-document relevance information for training the retrieval model. In fact, our model is also a framework for joint modeling of search and recommendation, where only user-item interaction data (i.e., recommendation data) is available.

3 METHODOLOGY

In this section, we formalize the problem definition and introduce the proposed joint search and recommendation framework (JSR). We further discuss the employed method for speeding up the training procedure. We additionally provide a matrix factorization interpretation of the model in order to find the connection between JSR and the recommender systems literature. We finally discuss item retrieval using the proposed model.

3.1 Problem Statement

Let $U = \{u_1, u_2, \cdots, u_m\}$ and $I = \{i_1, i_2, \cdots, i_n\}$ denote a set of m users and n items, respectively. Let $D = \{y_{ui} : u \in U, i \in I\}$ be a set of user-item interaction data, where y_{ui} represents the label corresponding to the interaction that the user u had with the item i. Labels can be either numeric (e.g., star rating) or binary (e.g., like/dislike). Instead of explicit feedback, y_{ui} can also be captured from user interactions with the recommender system as implicit feedback. Although implicit feedback can also be numeric (e.g., interaction count or time), binary labels are more common, such as clicking on a link, watching a movie, or listening to a music. Therefore, in this paper, we assume that labels are binary and were collected from implicit feedback, which is the most realistic setting in many applications [10, 15, 37]. Our framework can be easily extended to numeric labels as well.

In addition to user-item interactions, we assume that there is an additional set $I_T = \{t_1, t_2, \cdots, t_n\}$ containing a textual description for each item in I. These descriptions are often easy to collect, for example, from item descriptions, meta-data, or user-generated tags and reviews. Textual item description has been previously used for content-based and hybrid recommendation [5].

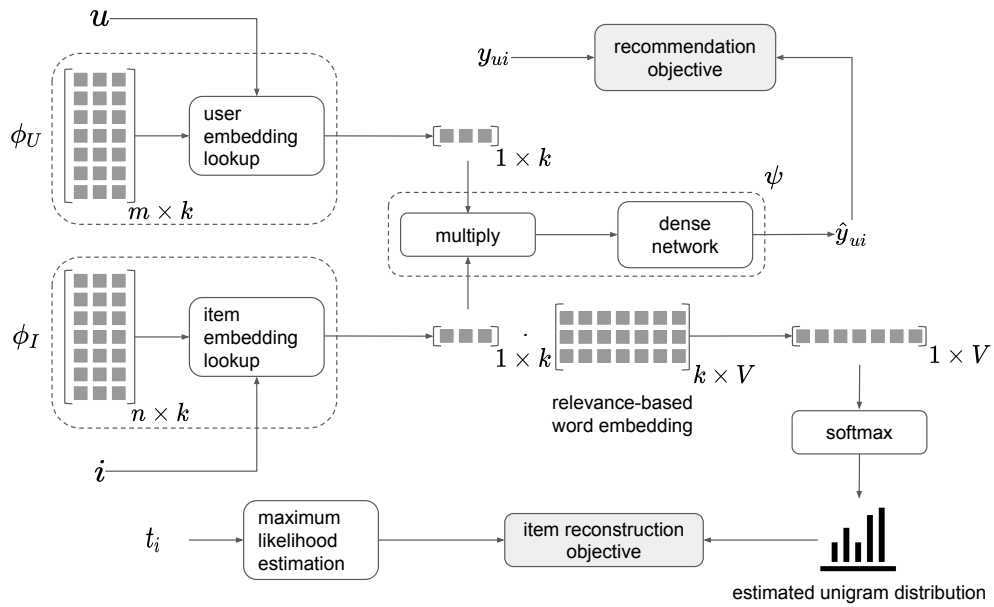

Figure 1: A high-level overview of the JSR framework that consists of three major components ϕ_U, ϕ_I, and ψ. JSR is trained using two objective functions: a recommendation objective and an item text reconstruction objective.

Given D and I_T, the goal is to develop a joint search-recommendation model. In more detail, the learned model should be able to (1) predict the future user-item interactions (i.e., recommendation) and (2) retrieve relevant items given a natural language query.

3.2 The JSR Framework

A natural implementation for joint modeling of search and recommendation is to optimize retrieval and recommendation objectives, simultaneously (similar to [52]). However, this is not possible without query-item relevance information. Therefore, JSR consists of the following two objectives: a recommendation objective and an item reconstruction objective. The high-level overview of JSR is depicted in Figure 1.

3.2.1 ***Recommendation.*** Our recommendation component is based on collaborative filtering which relies on user-item interactions. Collaborative filtering has shown to be effective in many recommendation scenarios. JSR estimates a recommendation score for each user-item pair as follows:

$$\hat{y}_{ui} = \psi(\phi_U(u), \phi_I(i)) \tag{1}$$

where $\hat{y}_{ui}$ is the model's prediction. The component ϕ_U (ϕ_I) learns a k-dimensional latent representation for each user (item). As demonstrated in Figure 1, we implement the recommendation component ψ by feeding the Hadamard product of the user and item representations to a fully-connected feed-forward neural network (called the dense network) with few hidden layers. Note that the Hadamard product is the element-wise multiplication of two matrices. We use ReLU as the activation function in the hidden layers and employ dropout in all hidden layers to avoid overfitting. The output activation in the dense network is a sigmoid function. The number of hidden layers, their sizes, and the dropout probability are hyper-parameters of the model.

We formulate the recommendation objective using a pointwise loss function. Following He et al. [15], we use a binary cross-entropy loss function that has shown effective performance in neural collaborative filtering for implicit data. As described in [15], the reason is that some popular loss functions in collaborative filtering, such as mean squared error, assume that the data is drawn from a Gaussian distribution, which does not hold in many settings [41]. In addition to the effectiveness of binary cross-entropy, it is a probabilistic loss derived from the log-likelihood maximization and can be easily combined with our second probabilistic objective. This loss function has been also used for training neural retrieval models in previous work [8, 20, 54].

Formally, let us first define b as a mini-batch of training data sampled from the training set D (see Section 3.1) expanded with η random negative samples per user-item interaction. η is a hyper-parameter. The loss function for a mini-batch b is defined as:

$$L_{bce} = -\frac{1}{|b|} \sum_{(u,i) \in b} y_{ui} \log \hat{y}_{ui} + (1 - y_{ui}) \log(1 - \hat{y}_{ui}) \tag{2}$$

We have also tried pairwise cross-entropy loss, however, no significant improvement has been observed in our experiments. Therefore, we have decided to keep the simpler pointwise model.

3.2.2 ***Item Text Reconstruction.*** The goal of item reconstruction is to make sure that the learned user and item representations can be mapped into a natural language space, and thus can be used for item retrieval. To this end, we use the textual descriptions of items (i.e., the set I_T; see Section 3.1 for more information). In more detail, we maximize the probability of generating the textual description of each item from the learned item representations. In this paper, we focus on unigram language model representation which has been shown to be effective in information retrieval [35]. In more detail, let $\mathcal{E} \in \mathbb{R}^{|V| \times k}$ denote a word embedding matrix with the same dimensionality as the user and item embeddings (k), where V is the vocabulary set. $\mathcal{E}$ is a pre-trained relevance-based word

embedding matrix [51]. Relevance-based word embedding models are based on the bag-of-words assumption and represent each word in a k-dimensional space to capture relevance information. The model is trained based on weak supervision and does not require any label data (see [8, 53] for more details). In fact, relevance-based word embedding models use the relevance models [24] as a weak supervision signal. We use relevance-based word embedding, since the goal of the model is to learn representations that are suitable for further retrieval purposes. For more information on relevance-based embedding, we refer the reader to [51]. Multiplication of each item embedding vector $\vec{i} \in \mathcal{I}$ to the transpose of the embedding matrix $\mathcal{E}$ results in a $|V|$-dimensional representation for item i. Therefore, our model estimates a unigram language model for item i as follows:

$$\theta_i = \text{softmax}(\mathcal{I}_{[i]}.\mathcal{E}^T) \tag{3}$$

The aim is to maximize the likelihood of generating the item description text. This is equivalent to minimizing the following cross-entropy for the mini-batch b:

$$L_{mlr} = -\frac{1}{|b|} \sum_{(u,i) \in b} \sum_{w \in t_i} \frac{\text{count}(w, t_i)}{|t_i|} \log p(w|\theta_i) \tag{4}$$

where $t_i \in I_T$ is a textual description for item i.

3.2.3 ***Optimization.*** We train the model using a gradient descent-based optimizer. The parameters that should be learned include the user embedding matrix $\mathcal{U}$, the item embedding matrix $\mathcal{I}$, and the parameters of the dense network for recommendation. Note that the embedding matrix $\mathcal{E}$ is pre-trained and fixed. We use Adam optimizer in our experiments to minimize the following loss function:

$$\mathcal{L} = L_{bce} + \alpha L_{mlr} \tag{5}$$

where α is a hyper-parameter controlling the weight of the item reconstruction loss.

3.3 Training Efficiency in JSR

Due to the large number of terms in a vocabulary (e.g., 500k), the unigram language model estimated by JSR is computationally expensive, because of the summation in the denominator of the softmax operator. Since this summation should be computed for every single item in the mini-batch at each training step, it substantially slows down the training process. To address this issue, we approximate the softmax operator using *hierarchical softmax*, which has been introduced by Morin and Bengio [32] for neural language modeling and successfully employed by Mikolov et al. [30] for word representation learning. This approximation uses a binary tree structure to represent vocabulary terms. Each leaf corresponds to a unique vocabulary term and there exists a unique path from the root of the tree to each leaf. This path is used for estimating the probability of the vocabulary term representing by the leaf. Since the height of the tree is $O(\log(|V|))$, the complexity of softmax calculation goes down from $O(|V|)$ to $O(\log(|V|))$. This results in a substantial improvement in computational complexity. We refer the reader to [31, 32] for the details of hierarchical softmax approximation.

3.4 Matrix Factorization Interpretation

Matrix factorization is the dominant approach in collaborative filtering. In this subsection, we briefly discuss how JSR can be interpreted as a matrix factorization model. For simplicity, assume that the dense network in our model (see Figure 1) is simply a linear regression model whose weights are set to 1. Therefore, feeding the Hadamard product of two vectors to this network is equivalent to their inner product. In this case, JSR can be modeled as:

$$\mathcal{U}^*, \mathcal{I}^* = \arg\min_{\mathcal{U}, \mathcal{I}} L_1(\mathcal{A}, \mathcal{U}\mathcal{I}^T) + \lambda L_2(\mathcal{T}, \mathcal{I}\mathcal{E}^T) \tag{6}$$

where $\mathcal{A} \in \mathbb{R}^{m \times n}$ denotes the user-item interaction matrix, containing the training data. Each row j of the matrix $\mathcal{T} \in \mathbb{R}^{n \times |V|}$ is a unigram language model for the item j, estimated using maximum likelihood estimation. L_1 is a recommendation loss that minimizes the user-item interaction reconstruction error, while L_2 is an item description reconstruction loss that minimizes the distance between the learned item representations and textual descriptions of the items. L_2 can also be seen as a *regularizer* that prevents the collaborative filtering model from overfitting. Item embedding has been previously used by Liang et al. [27] and by Train et al. [46] to regularize collaborative filtering, however, their models are based on item-item co-occurrences, which is different from ours.

Due to the non-linear operations in the dense network, JSR is the generalized version of this matrix factorization interpretation.

3.5 Item Retrieval using JSR

Once the model is trained, we compute the language model θ_i (see Equation (3)) for all items. To improve the retrieval efficiency, we take the top N vocabulary terms with the highest probability for each item (called $W_{\text{top}}(i)$) and normalize the language model as follows:

$$p(w|\hat{\theta}_i) = \begin{cases} \frac{p(w|\theta_i)}{\sum_{w' \in W_{\text{top}}(i)} p(w'|\theta_i)} & \text{if } w \in W_{\text{top}}(i) \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

We construct an inverted index from each word w in the vocabulary to a list of items as follows:

$$w \rightarrow \left\{ (i, p(w|\hat{\theta}_i)) : \forall i \in I \text{ such that } w \in W_{\text{top}}(i) \right\} \tag{8}$$

To compute the retrieval score for a natural language query q at the test time, we use a KL-divergence retrieval model [23] with Jelinek-Mercer smoothing:

$$\text{retrieval score}(q, i) = \sum_{w \in q} p(w|\theta_q) \log \left[\lambda p(w|\hat{\theta}_i) + (1 - \lambda) p(w|C) \right] \tag{9}$$

where $\lambda \in [0, 1]$ is the smoothing parameter and $p(w|C)$ denotes the probability of term in the collection, and can be computed as:

$$p(w|C) = \frac{1}{Z} \sum_{i \in I} p(w|\hat{\theta}_i) \tag{10}$$

where Z is a normalization factor. We develop two retrieval models based on different implementation of the query language model (θ_q). Similar to [35], the JSR-QL is our first retrieval model in which $p(w|\theta_q)$ is computed using maximum likelihood estimation (i.e., $p(w|\theta_q) = \frac{\text{count}(w,q)}{|q|}$). The second retrieval model, called JSR-RM, is a pseudo-relevance feedback algorithm based on the Lavrenko and Croft's relevance models [24]. In other words, we first retrieve documents using JSR-QL and then compute the relevance language model (i.e., RM3) and finally retrieve documents based on the re-estimated query language model. Refer to [1, 24] for the detailed implementation of RM3.

Table 1: Statistics of the data used in our experiments.

Data	# users	# items	# interactions	min interactions	sparsity	# queries (retrieval)
MovieLens 20M	138,493	27,278	20,000,263	20 per user	99.471%	919
Amazon - Electronics	192,403	63,001	1,689,188	5 per user	99.986%	989
Amazon - Kindle Store	68,223	61,934	989,618	5 per user	99.977%	4603
Amazon - Cell Phones & Accessories	27,879	10,429	194,439	5 per user	99.933%	165

3.6 Summary

The model optimizes two objectives simultaneously, one user-item interaction (recommendation) objective and one item text reconstruction objective. This allows the model to transfer knowledge from user-item interactions to the item representations. We use unigram language model for representing items and estimate it using a hierarchical softmax function which has shown to be highly efficient. The language model is obtained from the learned item representation multiplied by a relevance-based word embedding matrix which results in item language models that are suitable for retrieval purposes. Our matrix factorization interpretation of the framework shows that item text reconstruction can be seen as a regularization for the recommendation model.

4 EXPERIMENTS

In this section, we evaluate the proposed framework using a wide range of datasets.

4.1 Training

4.1.1 ***Training Data.*** We study the performance of our model on multiple public datasets. We use *MovieLens 20M* [13] which is the largest version of the MovieLens datasets to date and contains over 20 million total interactions with the minimum interaction of 20 per user.[1] MovieLens 20M is a standard dataset for evaluating collaborative filtering models in the context of movie recommendation. We also use three diverse product categories in the context of e-commerce. We adopt the five-core Amazon review dataset [14, 29], that covers user-item interactions (review, rating, helpfulness votes, etc) on 24 product categories from May 1996 to July 2014.[2] We have selected three product categories with different size and sparsity to observe the performance of the model on different conditions. They include Electronics, Kindle Store, and Cell Phones & Accessories. The largest category contains over 1.5 million interactions, while the number of interactions in the smallest category does not reach 200 thousands. The minimum number of interactions per user in these categories is 5. Similar to previous work [10, 15], we binarized the labels in all datasets by representing each user-item interaction with label 1 as an implicit feedback. Table 1 reports the statistics of the datasets.

To construct the set I_T (see Section 3.1), we concatenated the tags that users have provided for each movie in the MovieLens 20M dataset. For the Amazon datasets, we selected the most helpful review of each item (according to their helpfulness scores provided by users) as its textual description. If there was no review with positive helpfulness score, we randomly selected one of the reviews. We cleaned up the data by removing non-alphabetic characters and stopwords from item descriptions.

[1]MovieLens 20M can be found at https://grouplens.org/datasets/movielens/20m/.
[2]The Amazon review dataset can be found at http://jmcauley.ucsd.edu/data/amazon/.

Following previous work [15, 26, 37], we adopt a leave-one-out evaluation methodology for evaluation. For each user, we held-out the latest interaction as the test recommendation data and utilized the remaining data for training.

4.1.2 ***Parameter Setting.*** We implemented our model using TensorFlow.[3] In all experiments, the network parameters were optimized using the Adam optimizer. For hyper-parameter optimization, we selected the latest interaction of each user in the training set as a validation set. We performed grid search and chose the hyper-parameters based on the loss values obtained on the validation set. After the hyper-parameter selection process, we train the model with the chosen hyper-parameter values on the original training set. The learning rate was selected from $\{1 \times 10^{-5}, 5 \times 10^{-4}, 1 \times 10^{-4}, 5 \times 10^{-4}, 1 \times 10^{-3}\}$. The batch size was selected from $\{32, 64, 128, 256, 512\}$. The dropout keep probability was selected from $\{0.7, 0.8, 0.9, 1.0\}$. The dimensionality of word, user, and item embedding vectors were set to 50 and the word embedding matrix was initialized by a relevance-based word embedding model, called relevance likelihood maximization (RLM) [51]. The embedding vectors were trained using the ClueWeb collection as described in [51]. We set the number of negative samples per interaction (i.e., parameter η) to 4. The number of hidden layers in the dense network and their output sizes were selected from $\{1, 2, 3\}$ and $\{10, 20, 50, 100\}$, respectively.

4.2 Evaluating the Retrieval Results

4.2.1 ***Evaluation Data.*** This experiment is challenging, since, to the best of our knowledge, there is no public dataset containing both user-item interactions and query-item relevance information on the same item set. Therefore, we created a dataset for evaluating the model trained on the MovieLens 20M dataset, and used an automatic evaluation methodology employed by [2, 48, 52] for evaluating the models trained on the Amazon datasets.

Movie retrieval dataset: We adopt the known-item search data created by Hagen et al. [12]. The data contains difficult real-world information needs collected from Yahoo! Answers. From this data, we only selected the information needs with the category "Movies". The relevance judgments contain a single relevant document per information need. The relevant documents were selected from the ClueWeb collection. By manual annotation, we linked each relevant document in the dataset to the corresponding movie ID in the MovieLens 20M dataset. We finally filtered out the queries whose answers were not found in MovieLens 20M. This results in 919 queries, written by real users in the Yahoo! Answers website, each has exactly one relevant movie from the MovieLens 20M dataset. We refer the reader to [12] for some sample queries. To foster research in this area, we have made our manual annotations publicly available.[4]

[3]https://www.tensorflow.org/
[4]The dataset is available at http://ciir.cs.umass.edu/downloads/movie-search-ml20/.

Table 2: Retrieval performance of JSR and the baselines. The highest value per column is marked in bold, and the superscript * denotes statistically significant improvements compared to all the baselines.

	MovieLens 20M		Amazon - Electronics		Amazon - Kindle Store		Amazon - Cell Phones	
Model	**MRR**	**nDCG@10**	**MAP**	**nDCG@10**	**MAP**	**nDCG@10**	**MAP**	**nDCG@10**
QL	0.033	0.036	0.372	0.421	0.181	0.210	0.236	0.267
BM25	0.030	0.032	0.351	0.408	0.185	0.219	0.244	0.275
RM3	0.035	0.036	0.386	0.443	0.193	0.219	0.256	0.286
ERM	0.058	0.063	0.400	0.458	0.213	0.235	0.284	0.312
PRP+	0.094	0.112	0.431	0.511	0.251	0.297	0.328	0.370
PRP+ & ERM	0.102	0.137	0.457	0.560	0.306	0.366	0.347	0.381
JSR-QL	0.121*	0.163*	0.511*	0.608*	0.348*	0.396*	0.366*	0.408*
JSR-RM	**0.133***	**0.169***	**0.518***	**0.614***	**0.362***	**0.407***	**0.381***	**0.425***

Product retrieval dataset: The Amazon product data does not contain search queries, thus cannot be directly used for evaluating retrieval models. As Rowley [40] investigated, directed product search queries contain either a producer's name, a brand, or a set of terms describing the product category. Following this observation, Van Gysel et al. [48] proposed to automatically generate queries based on the product categories. To be concise, for each item in a category c, a query q is generated based on the terms in the category hierarchy of c. Then, all the items within that category are marked as relevant for the query q. The detailed description of the query generation process can be found in [2]. Although the queries in this data were automatically constructed, since the query generation process has been done based on observations from real user queries, it has become a standard approach for evaluating product search, and has been used by the research community [2, 11, 48, 52].

4.2.2 ***Evaluation Metrics.*** In the movie retrieval dataset, there is only one relevant item per query. This characteristic makes some of the common IR evaluation metrics unsuitable. For this dataset, we use mean reciprocal rank (MRR) and normalized discounted cumulative gain (nDCG) [21] of the top 10 items (nDCG@10). However, in the Amazon datasets, there exist multiple relevant items per query. In such cases, mean average precision (MAP) and nDCG@10 are used as evaluation metrics.

Statistically significant differences of performances were computed using the two-tailed paired t-test with Bonferroni correction at a 99% confidence level.

4.2.3 ***Experimental Setup.*** The hyper-parameters of all the retrieval baselines as well as the proposed model were selected using two-fold cross-validation over queries. In hyper-parameter optimization, the regularization parameter α and the smoothing parameter λ were selected from $(0, 1)$.

4.2.4 ***Results and Discussion.*** We compare the retrieval performance of the proposed method to the following baselines:

Query Likelihood (QL) [35]: This is a language model-based retrieval model that uses the Dirichlet prior method [56] for document language model smoothing.

BM25 [39]: This is a simple yet effective probabilistic retrieval model derived from a 2-Poisson distribution approximation for document modeling.

Relevance Model (RM3) [24]: This is an effective pseudo-relevance feedback model based on the language modeling framework that uses the top retrieved documents for query expansion.

Embedding-based Relevance Model (ERM) [50]: This is a state-of-the-art pseudo-relevance feedback method that takes advantage of pre-trained word embedding vectors. Similar to JSR, ERM also uses the relevance-based word embedding [51] which has shown superior performance to models like word2vec [30] and GloVe [34].

Since the authors are not aware of any retrieval baseline that use user-item interactions, we adapt the probabilistic relevance propagation (PRP) of Shakery and Zhai [44]. PRP uses hyperlinks in the Web for retrieval score estimation and language model smoothing [45]. Based on the user-item interaction matrix, we constructed a graph of users and items and applied the PRP algorithm. This model utilizes user-item interaction information in retrieval. We call the model PRP+. We also enhance this model by using embedding-based relevance model (ERM) [50] as for pseudo-relevance feedback (called PRF+ & ERM).

Each baseline has a number of hyper-parameters, such as smoothing parameter (μ), the number of feedback terms, the feedback coefficient, etc. As mentioned earlier, we tune the hyper-parameters using two-fold cross-validation over the retrieval queries. In all the baselines, we used the same textual descriptions used in training our model (see Section 4.1 for more information). To have a fair evaluation, we do not consider supervised retrieval models, including the joint model of Zamani and Croft [52], in our baselines, because our model does not also use any query-document relevance signal.

The retrieval performance of the methods are presented in Table 2. The results indicate the difficulty of the movie retrieval dataset compared to the others. This is due to the nature of the queries and item descriptions; queries are long and descriptive, while item descriptions are short. RM3 does not perform much better the QL, due to the length of documents and the low performance of the first stage retrieval model (i.e., QL). Comparing the results obtained by the JSR and PRP+ models against the other baselines demonstrates the importance of user-item interaction for retrieval tasks. According to the table, both JSR models outperform competitive well-tuned retrieval baselines. The improvements are statistically significant in all cases. Moreover, JSR-RM achieves the best retrieval results. The highest performance is achieved on the Amazon - Electronics dataset, which is the largest product dataset, in terms of the number of user-item interactions.

4.3 Evaluating the Recommendation Results

4.3.1 ***Evaluation Data.*** We evaluate the recommendation performance based on a leave-one-out strategy. In more detail, the last interaction of each user was chosen as a test data. We further followed the common strategy of taking 100 random negative sample items per user, as been widely used in the literature [10, 15, 22].

Table 3: Recommendation performance of JSR and the baselines. The highest value per column is marked in bold, and the superscript * denotes statistically significant improvements compared to all the collaborative filtering baselines (i.e., ItemPopularity, BPR, eALS, and NCF).

	MovieLens 20M		Amazon - Electronics		Amazon - Kindle Store		Amazon - Cell Phones	
Model	**nDCG**	**HR**	**nDCG**	**HR**	**nDCG**	**HR**	**nDCG**	**HR**
ItemPopularity	0.5226	0.8196	0.3227	0.5044	0.1875	0.3340	0.2214	0.3766
BPR	0.6086	0.8633	0.4820	0.7639	0.3881	0.5845	0.3692	0.5038
eALS	0.6171	0.8820	0.4906	0.7834	0.3863	0.5809	0.3746	0.5152
NCF	0.6247	0.9050	0.4841	0.7624	0.3910	0.5999	0.3775	0.5203
CDL	0.6321	0.9183	0.5081	0.8021	0.4310	0.6570	0.4112	0.5898
CF-aSDAE	0.6319	0.9177	0.5109	0.8073	0.4285	0.6503	**0.4183**	0.6022
JSR	**0.6345***	**0.9210***	**0.5125***	**0.8100***	**0.4335***	**0.6627***	0.4176*	**0.6085***

4.3.2 ***Evaluation Metrics.*** To evaluate the recommendation performance, we use normalized discounted cumulative gain (nDCG) [21] and hit ratio (HR). The cut-off for these metrics is set to 10. HR measures whether the test item is present on the top 10 list. Note that since there is only one relevant item per user in the leave-one-out strategy, HR is equivalent to recall. On the other hand, nDCG is a ranking metric accounting for the position of the hit by assigning higher scores to hits at top ranks. Although nDCG is often used for evaluating items with graded relevance labels, we use this metric to have our results comparable with previous work [10, 15]. We calculated both metrics for each test user and reported the average score. Similar to the retrieval experiments, statistically significant differences of performances were computed using the two-tailed paired t-test with Bonferroni correction at a 99% confidence level.

4.3.3 ***Results and Discussion.*** We compare the effectiveness of JSR to the following collaborative filtering and hybrid baselines:

- ItemPopularity: This simple baseline computes the popularity of each item based on the number of interactions on the item in the training set. This is a non-personalized method to benchmark the recommendation performance.
- Bayesian Personalized Ranking (BPR) [37]: This is a competitive matrix factorization method, adapted for learning from implicit feedback. BPR takes advantage of a pairwise ranking loss.
- Element-wise Alternating Least Squares (eALS) [16]: This method is an effective factorization method for item recommendation with implicit feedback. This baseline takes all unobserved items as negative instances and weights them based on their popularity. It outperforms weighted matrix factorization (WMF) [19] that uses a uniform weighting for negative instances.
- Neural Collaborative Filtering (NCF) [15]: This neural network baseline is a combination of a generalized matrix factorization and a fully-connected network that uses a cross-entropy loss function for collaborative filtering with implicit feedback.
- Collaborative Deep Learning (CDL) [49]: This neural network hybrid recommendation model jointly performs deep representation learning for the content information and collaborative filtering for the user-item interactions.
- Collaborative Filtering with Additional Stacked Denoising Autoencoders (CF-aSDAE) [9]: This hybrid recommendation model uses stacked denoising autoencoders to jointly model deep users and items' latent factors from side information and collaborative filtering from the interaction matrix.

Table 4: The top 10 words selected by JSR for four sample movies. The words are sorted in descending order in terms of their weights. The table should be viewed in color.

The Lord of the Rings (1978)	Batman Returns (1992)	Gandhi (1982)	The Mask (1994)
fantasy	batman	documentary	cartoon
magic	character	film	parody
movies	superhero	directed	movie
wizard	horror	prize	black[5]
animation	thriller	award	comic
potter	starring	supporting	comedy
cartoon	fantasy	films	film
fiction	movie	movie	monster
classic	joker	fiction	thriller
novel	comedy	drama	shows

Each baseline has a number of hyper-parameters, such as learning rate and the number of latent factors. We tune all of these hyper-parameters using the same procedure as the proposed method using the same validation data (see Section 4.1.2).

Table 3 reports the recommendation performance achieved by JSR and the baselines. According to the table, BPR, eALS, and NCF are all competitive collaborative filtering baselines, and there is no clear winner among them. NCF outperforms the other baselines in three datasets (MovieLens 20M, Amazon-Kindle Store, and Amazon-Cell Phones), while eALS shows superior performance among the baselines in the Amazon-Electronics data. This might be due to the sparsity of the collections, i.e., Amazon-Electronics is the sparsest dataset (see Table 1). The eALS baseline takes negative samples based on the popularity of items, while the others use a uniform sampling method. The results also show that JSR significantly outperforms all the collaborative filtering baselines in all the datasets. Our model also shows a comparable performance (and in most cases slightly better) performance to the hybrid recommendation baselines. This indicates that although JSR only uses the item descriptions for regularization, it effectively takes advantage of side information.

4.4 Additional Analysis

In this section, we provide additional empirical analysis to have a better understanding the model's performance. We first do a case study by looking at the top terms chosen by the model for

[5]The Mask is considered as a "black comedy".

Table 5: Corpora used for training the embedding vectors.

ID	corpus	# tokens
GloVe - Wiki	Wikipedia 2014 & Gigawords 5	6b
GloVe - 840b	Web crawl	840b
GloVe - ClueWeb	Web crawl	70b

a few items. We further investigate the impact of different word embedding vectors on the model's performance.

4.4.1 ***Analyzing the Learned Representations.*** To provide a deeper analysis on the quality of the learned representation, we report the top 10 terms with the highest weight in the learned representations for four sample movies (for the model trained on the MovieLens 20M data) in Table 4. In the table, we classify each word into one of the following categories:

- Green: related to the movie genre
- Gray: related to the movie content (e.g., characters and scenes)
- Blue: related to other movies similar to the movie
- Yellow: miscellaneous features (e.g., awards, etc.)
- Red: not directly related to the movie
- White: general terms (e.g., movie and film)

There are some common words that appear in most movie representations, such as "movie", "film", "films". In fact, due to the nature of the MovieLens 20M dataset, these terms are considered as general terms. There exist more than one green cell in each column of the table, which indicates that JSR pays attention to the genre of the movies. The movie Gandhi has won eight Academy Awards and interestingly, JSR gives high weights to the words such as "prize" and "award" (yellow cells). This might be due to the fact that many users who watched Gandhi were interested in award-winning movies. Surprisingly, we observe the term "potter" among the top 10 terms for the movie The Lord of the Rings. This is most likely selected because many users who watched The Lord of the Rings also watched the Harry Potter movies. In addition, there are some terms that we do not have direct explanations for. For example, it is not clear why the term "comedy" is selected for the movie Batman Returns. They might be due to the user interactions with other movies. They might also be due to the model's errors. The dark comedy aspect of this movie could be also the reason.

4.4.2 ***Investigating the Impact of Word Embedding Vectors.*** As mentioned multiple times in the paper, our model uses relevance-based word embedding. In this section, we investigate the performance of the model when using general-purpose word embedding vectors, such as GloVe [34]. Table 6 shows the results for the model with various word embedding matrices. For the sake of space, in this experiments, we report the result for all the Amazon datasets together. The description of the training corpora is presented in Table 5. According to the results, relevance-based word embedding (RLM) leads to significantly better retrieval results. The reason is that the objective in such models were specifically designed for information retrieval purposes. Note that none of these word embedding models require label data for training.

5 POTENTIAL APPLICATIONS

In this section, we review a number of potential applications in which the proposed framework can be potentially useful.

Interpretability: Interpretation in machine learning helps researchers and engineers identify what has been learned by the model, what biases have affected the model, and how to improve the performance. It is especially desired in collaborative filtering models, since they heavily rely on latent features. Since our model learns representations that can be mapped to a unigram language model, it can be potentially used for improving the interpretability of filtering models.

Transparency and Explainability: Users also prefer to receive an explanation for recommendations [58]. In addition, explainability improves the transparency of the system. Learning user and item representations that can be mapped to a natural language space could potentially ease the process of explaining recommendations.

User Profiling: Table 4 shows that JSR learns interpretable representations for items. Therefore, it can also be used for user profiling; representing the user's interests with natural language. The system can also allow the user to manually modify the learned natural language representation for improving the recommendation.

Universal Representation Across Domain and Modality: Natural language is a universal representation; meaning that each movie, music, image, book, etc. can be represented using natural language. Mapping multi-domain item representations to a natural language space would potentially close the gap between these domains and modalities.

Conversational Recommendation: Natural language is the most convenient interaction for human. In conversational recommender systems, users describe their interests in order to receive accurate recommendations. Learning natural language representations for items could be of potential use in conversational recommendation.

Table 6: Retrieval performance of JSR with different word embedding initialization. The highest value per column is marked in bold, and the superscript * denotes statistically significant improvements compared to all the baselines.

	MovieLens 20M		Amazon - All	
Embedding	**MRR**	**nDCG**	**MAP**	**nDCG**
GloVe - Wiki	0.088	0.103	0.342	0.386
GloVe - 820b	0.104	0.124	0.358	0.415
GloVe - ClueWeb	0.096	0.116	0.351	0.408
RLM	**0.133***	**0.169***	**0.389***	**0.443***

6 CONCLUSIONS AND FUTURE WORK

This paper introduced the notion of learning a retrieval model from user-item interaction data. This has multiple potential applications in real-world scenarios, such as developing an accurate retrieval model for a system that has already collected rich user-item interaction data. This includes media streaming services, such as Netflix and Spotify, and e-commerce websites. We presented a model that predicts user-item interactions (similar to collaborative filtering models) and reconstructs the item description text using multi-task learning. The proposed model is based on neural networks and relevance-based word embeddings and does not require any query-item relevance information for training. Our experiments on four diverse datasets demonstrated that our model substantially outperforms competitive baselines in terms of retrieval performance. In addition, the recommendation performance of the model is significantly higher than that of the state-of-the-art collaborative filtering models and is comparable with the hybrid recommendation models.

Our analysis demonstrated that the learned representations took advantage of the collaborative data for item representation.

In addition to exploring potential applications mentioned in Section 5, we intend to study the joint search and recommendation model, where limited retrieval training data (query-item relevance information) is available. In the future, we will also study joint search and recommendation models in an online setting.

Acknowledgements. This work was supported in part by the Center for Intelligent Information Retrieval and in part by NSF IIS-1715095. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsor. The authors would like to thank Qingyao Ai, Mohammad Aliannejadi, Helia Hashemi, and Youngwoo Kim for their invaluable feedback.

REFERENCES

[1] Nasreen Abdul-jaleel, James Allan, W. Bruce Croft, Fernando Diaz, Leah Larkey, Xiaoyan Li, Donald Metzler, Mark D. Smucker, Trevor Strohman, Howard Turtle, and Courtney Wade. 2004. UMass at TREC 2004: Novelty and HARD. In *TREC*.

[2] Qingyao Ai, Yongfeng Zhang, Keping Bi, Xu Chen, and W. Bruce Croft. 2017. Learning a Hierarchical Embedding Model for Personalized Product Search. In *SIGIR '17*. 645–654.

[3] Trapit Bansal, Mrinal Das, and Chiranjib Bhattacharyya. 2015. Content Driven User Profiling for Comment-Worthy Recommendations of News and Blog Articles. In *RecSys '15*. 195–202.

[4] Nicholas J. Belkin and W. Bruce Croft. 1992. Information Filtering and Information Retrieval: Two Sides of the Same Coin? *Commun. ACM* 35, 12 (Dec. 1992), 29–38.

[5] Robin Burke. 2002. Hybrid Recommender Systems: Survey and Experiments. *User Modeling and User-Adapted Interaction* 12, 4 (Nov. 2002), 331–370.

[6] Ching-Wei Chen, Paul Lamere, Markus Schedl, and Hamed Zamani. 2018. Recsys Challenge 2018: Automatic Music Playlist Continuation. In *RecSys '18*. 527–528.

[7] Bruce Croft, Donald Metzler, and Trevor Strohman. 2009. *Search Engines: Information Retrieval in Practice* (1st ed.). Addison-Wesley Publishing Company.

[8] Mostafa Dehghani, Hamed Zamani, Aliaksei Severyn, Jaap Kamps, and W. Bruce Croft. 2017. Neural Ranking Models with Weak Supervision. In *SIGIR '17*. 65–74.

[9] Xin Dong, Lei Yu, Zhonghuo Wu, Yuxia Sun, Lingfeng Yuan, and Fangxi Zhang. 2017. A Hybrid Collaborative Filtering Model with Deep Structure for Recommender Systems. In *AAAI '17*. 1309–1315.

[10] Travis Ebesu, Bin Shen, and Yi Fang. 2018. Collaborative Memory Network for Recommendation Systems. In *SIGIR '18*. 515–524.

[11] Yangyang Guo, Zhiyong Cheng, Liqiang Nie, Yinglong Wang, Jun Ma, and Mohan Kankanhalli. 2019. Attentive Long Short-Term Preference Modeling for Personalized Product Search. *ACM Trans. Inf. Syst.* 37, 2 (2019), 19:1–19:27.

[12] Matthias Hagen, Daniel Wägner, and Benno Stein. 2015. A Corpus of Realistic Known-Item Topics with Associated Web Pages in the ClueWeb09. In *ECIR '15*. 741–754.

[13] F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. *ACM Trans. Interact. Intell. Syst.* 5, 4 (2015), 19:1–19:19.

[14] Ruining He and Julian McAuley. 2016. Ups and Downs: Modeling the Visual Evolution of Fashion Trends with One-Class Collaborative Filtering. In *WWW '16*. 507–517.

[15] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. In *WWW '17*. 173–182.

[16] Xiangnan He, Hanwang Zhang, Min-Yen Kan, and Tat-Seng Chua. 2016. Fast Matrix Factorization for Online Recommendation with Implicit Feedback. In *SIGIR '16*. 549–558.

[17] Will Hill, Larry Stead, Mark Rosenstein, and George Furnas. 1995. Recommending and Evaluating Choices in a Virtual Community of Use. In *CHI '95*. 194–201.

[18] Liang Hu, Songlei Jian, Longbing Cao, and Qingkui Chen. 2018. Interpretable Recommendation via Attraction Modeling: Learning Multilevel Attractiveness over Multimodal Movie Contents. In *IJCAI '18*. 3400–3406.

[19] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative Filtering for Implicit Feedback Datasets. In *ICDM '08*. 263–272.

[20] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning Deep Structured Semantic Models for Web Search Using Clickthrough Data. In *CIKM '13*. 2333–2338.

[21] Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated Gain-based Evaluation of IR Techniques. *ACM Trans. Inf. Syst.* 20, 4 (Oct. 2002), 422–446.

[22] Yehuda Koren. 2008. Factorization Meets the Neighborhood: A Multifaceted Collaborative Filtering Model. In *KDD '08*. 426–434.

[23] John Lafferty and Chengxiang Zhai. 2001. Document Language Models, Query Models, and Risk Minimization for Information Retrieval. In *SIGIR '01*. 111–119.

[24] Victor Lavrenko and W. Bruce Croft. 2001. Relevance Based Language Models. In *SIGIR'01*. 120–127.

[25] Hang Li. 2011. *Learning to Rank for Information Retrieval and Natural Language Processing*. Morgan & Claypool Publishers.

[26] Sheng Li, Jaya Kawale, and Yun Fu. 2015. Deep Collaborative Filtering via Marginalized Denoising Auto-encoder. In *CIKM '15*. 811–820.

[27] Dawen Liang, Jaan Altosaar, Laurent Charlin, and David M. Blei. 2016. Factorization Meets the Item Embedding: Regularizing Matrix Factorization with Item Co-occurrence. In *RecSys '16*. 59–66.

[28] Nicolaas Matthijs and Filip Radlinski. 2011. Personalizing Web Search Using Long Term Browsing History. In *WSDM '11*. 25–34.

[29] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton van den Hengel. 2015. Image-Based Recommendations on Styles and Substitutes. In *SIGIR '15*. 43–52.

[30] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *NeurIPS '13*. 3111–3119.

[31] Andriy Mnih and Geoffrey E Hinton. 2009. A Scalable Hierarchical Distributed Language Model. In *NeurIPS '09*. 1081–1088.

[32] Frederic Morin and Yoshua Bengio. 2005. Hierarchical Probabilistic Neural Network Language Model. In *AISTATS '05*. 246–252.

[33] Michael G. Noll and Christoph Meinel. 2007. Web Search Personalization Via Social Bookmarking and Tagging. In *ISWC '07*. 367–380.

[34] Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global Vectors for Word Representation. In *EMNLP '14*. 1532–1543.

[35] Jay M. Ponte and W. Bruce Croft. 1998. A Language Modeling Approach to Information Retrieval. In *SIGIR'98*. 275–281.

[36] Hossein Rahmatizadeh Zagheli, Hamed Zamani, and Azadeh Shakery. 2017. A Semantic-Aware Profile Updating Model for Text Recommendation. In *RecSys '17*. 316–320.

[37] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian Personalized Ranking from Implicit Feedback. In *UAI '09*. 452–461.

[38] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. 1994. GroupLens: An Open Architecture for Collaborative Filtering of Netnews. In *CSCW '94*. 175–186.

[39] S. E. Robertson, E. Walker, S. Jones, M. M. Hancock-Beaulieu, and M. Gatford. 1994. Okapi at TREC-3. In *TREC*.

[40] Jennifer Rowley. 2000. Product Search in e-Shopping: A Review and Research Propositions. *Journal of Consumer Marketing* 17, 1 (2000), 20–35.

[41] Ruslan Salakhutdinov and Andriy Mnih. 2007. Probabilistic Matrix Factorization. In *NeurIPS '07*. 1257–1264.

[42] Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey Hinton. 2007. Restricted Boltzmann Machines for Collaborative Filtering. In *ICML '07*. 791–798.

[43] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based Collaborative Filtering Recommendation Algorithms. In *WWW '01*. 285–295.

[44] Azadeh Shakery and ChengXiang Zhai. 2006. A Probabilistic Relevance Propagation Model for Hypertext Retrieval. In *CIKM '06*. 550–558.

[45] Azadeh Shakery and Chengxiang Zhai. 2008. Smoothing Document Language Models with Probabilistic Term Count Propagation. *Inf. Retr.* 11, 2 (2008), 139–164.

[46] Thanh Tran, Kyumin Lee, Yiming Liao, and Dongwon Lee. 2018. Regularizing Matrix Factorization with User and Item Embeddings for Recommendation. In *CIKM '18*. 687–696.

[47] Yury Ustinovskiy and Pavel Serdyukov. 2013. Personalization of Web-search Using Short-term Browsing Context. In *CIKM '13*. 1979–1988.

[48] Christophe Van Gysel, Maarten de Rijke, and Evangelos Kanoulas. 2016. Learning Latent Vector Spaces for Product Search. In *CIKM '16*. 165–174.

[49] Hao Wang, Naiyan Wang, and Dit-Yan Yeung. 2015. Collaborative Deep Learning for Recommender Systems. In *KDD '15*. 1235–1244.

[50] Hamed Zamani and W. Bruce Croft. 2016. Embedding-based Query Language Models. In *ICTIR '16*. 147–156.

[51] Hamed Zamani and W. Bruce Croft. 2017. Relevance-based Word Embedding. In *SIGIR '17*. 505–514.

[52] Hamed Zamani and W. Bruce Croft. 2018. Joint Modeling and Optimization of Search and Recommendation. In *DESIRES '18*. 36–41.

[53] Hamed Zamani and W. Bruce Croft. 2018. On the Theory of Weak Supervision for Information Retrieval. In *ICTIR '18*. 147–154.

[54] Hamed Zamani, Mostafa Dehghani, W. Bruce Croft, Erik Learned-Miller, and Jaap Kamps. 2018. From Neural Re-Ranking to Neural Ranking: Learning a Sparse Representation for Inverted Indexing. In *CIKM '18*. 497–506.

[55] Hamed Zamani, Markus Schedl, Paul Lamere, and Ching-Wei Chen. 2019. An Analysis of Approaches Taken in the ACM RecSys Challenge 2018 for Automatic Music Playlist Continuation. *ACM Trans. Intell. Syst. Technol.* 10, 5 (2019).

[56] Chengxiang Zhai and John Lafferty. 2004. A Study of Smoothing Methods for Language Models Applied to Information Retrieval. *ACM Trans. Inf. Syst.* 22, 2 (2004), 179–214.

[57] Yongfeng Zhang, Qingyao Ai, Xu Chen, and W. Bruce Croft. 2017. Joint Representation Learning for Top-N Recommendation with Heterogeneous Information Sources. In *CIKM '17*. 1449–1458.

[58] Yongfeng Zhang and Xu Chen. 2018. Explainable Recommendation: A Survey and New Perspectives. *CoRR* abs/1804.11192 (2018).