Discrete Deep Learning for Fast Content-Aware Recommendation

Yan Zhang, Hongzhi Yin[†*], Zi Huang[†], Xingzhong Du[†], Guowu Yang, Defu Lian[*]
School of Computer Science and Engineering, University of Electronic Science and Technology of China
[†]School of Information Technology and Electrical Engineering, The University of Queensland
yixianqianzy@gmail.com,h.yin1@uq.edu.au,huang@itee.uq.edu.au,x.du@uq.edu.au,guowu@uestc.edu.cn,dove.ustc@gmail.com

ABSTRACT

Cold-start problem and recommendation efficiency have been regarded as two crucial challenges in the recommender system. In this paper, we propose a hashing based deep learning framework called Discrete Deep Learning (DDL), to map users and items to Hamming space, where a user's preference for an item can be efficiently calculated by Hamming distance, and this computation scheme significantly improves the efficiency of online recommendation. Besides, DDL unifies the user-item interaction information and the item content information to overcome the issues of data sparsity and cold-start. To be more specific, to integrate content information into our DDL framework, a deep learning model, Deep Belief Network (DBN), is applied to extract effective item representation from the item content information. Besides, the framework imposes balance and irrelevant constraints on binary codes to derive compact but informative binary codes. Due to the discrete constraints in DDL, we propose an efficient alternating optimization method consisting of iteratively solving a series of mixed-integer programming subproblems. Extensive experiments have been conducted to evaluate the performance of our DDL framework on two different Amazon datasets, and the experimental results demonstrate the superiority of DDL over the state-of-the-art methods regarding online recommendation efficiency and cold-start recommendation accuracy.

KEYWORDS

recommender system; Deep Learning; Hash code; Cold-start

ACM Reference Format:
Yan Zhang, Hongzhi Yin[†][1], Zi Huang[†], Xingzhong Du[†], Guowu Yang, Defu Lian[1]. 2018. Discrete Deep Learning for Fast Content-Aware Recommendation. In *WSDM 2018: 11th ACM International Conference on Web Search and Data Mining, February 5–9, 2018, Marina Del Rey, CA, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3159652.3159688

*The corresponding authors

WSDM 2018, February 5–9, 2018, Marina Del Rey, CA, USA

ACM ISBN 978-1-4503-5581-0/18/02…$15.00
https://doi.org/10.1145/3159652.3159688

1 INTRODUCTION

In the era of information explosion, information overload becomes a challenging problem. Thus it is essential to sort out valuable information for everyone. Personalized recommender systems have been recognized as one of the most critical and effective approaches for alleviating information overload. It is a key factor for the success of many online e-commerce websites such as Taobao, Amazon, Netflix, Yelp, etc.

Collaborative filtering (CF) based recommender systems proven to be very successful. They produce top-k items that users may be interested in by exploiting the historical interaction data such as ratings, purchasing, clicking, and watching records. Among all CF-based methods, the latent factor models (e.g., matrix factorization) have been demonstrated to achieve great success in both academia and industry. Such CF methods factorize an $n \times m$ user-item interaction matrix into a low-dimensional latent vector (a.k.a. feature) space where both users and items are represented by real-valued vectors. Then the user's preference scores for items were predicted by inner products between their vector representations, and the user's top-k preferred items can be produced by ranking the scores descendingly.

However, the growing scale of users and items has made online recommendation much more challenging. Specifically, suppose there are n users and m items in an online recommender system, users and items are denoted as r-dimension real-valued vectors, searching the top-k items that users may interested in is the task of online recommendation. The time complexity of online recommendation is $\mathcal{O}(nmr + nm \log k)$ [36], and it is a critical efficiency bottleneck when the size of the dataset is large. Hence a few recommendation frameworks [27–33] were proposed to speed up the online recommendation. However, the time complexity was not decreased markedly since these recommendations are still based on real-valued features.

Fortunately, hashing technique, encoding users and items into binary codes in Hamming space, is a promising approach to tackle the efficiency bottleneck. Since preference score, in this case, can be efficiently computed by bit operations, i.e., Hamming distance. One can even use a fast and accurate indexing method to find approximate top-k preferred items with sublinear or logarithmic time complexity [25]. Besides, each dimension of hash codes can be stored by only one

bit instead of 32/64 bits that are used for storing one dimension in real-valued vectors, which significantly reduces storage cost. However, many previous hashing-based recommendation improves the recommendation efficiency at the price of recommendation accuracy due to a large amount of information loss caused by discretization.

Existing hashing-based recommender systems were proposed to find some trade-off strategies between efficiency and accuracy [35–39]. These strategies consist of two types: two-stage hashing recommendation and hashing learning recommendation. To learn hash codes, a discrete optimization problem is formulated in hashing-based recommendation frameworks. However, this discrete optimization problem is NP-hard [6], but can be resorted to a two-stage procedure, which consists of relaxed optimization via discarding the discrete constraints, and subsequent binary quantization stage. But these two-stage approaches oversimplify original discrete optimization, so two principle hashing-based frameworks called Discrete Collaborative Filtering (DCF) [35] and Discrete Personalized Ranking (DPR) [36], were proposed to solve the discrete optimization directly. However, hash codes intuitively carry less information than real-valued vectors since much fewer bits are utilized to store information, which leads to poor recommendation performance. Besides, it is widely recognized that CF-based methods suffer poor performance when the interaction information is sparse, and an extreme case is cold start [18], where there is no interaction information between new users/items and other items/users. Because the above hashing recommendation frameworks are based on CF, data sparsity further aggravates the poor performance, and they cannot work in the cold-start setting.

To alleviate the above issues, we propose a hashing-based learning recommendation framework, Discrete Deep Learning (DDL), which combines a deep learning framework, Deep Belief Network (DBN), and CF framework to learn efficient binary representations for users and items from rating and content data. In order to obtain compact and informative hash codes, we add balance and irrelevant constraints on binary codes. Two steps solve the problem: initialize DDL and optimize DDL. We first initialize DDL by the *pre-train* procedure of DBN that will be introduced in Section 3.3. We then apply an efficient alternating optimization method to optimize our proposed DDL to obtain hash codes for users and items from rating and content data. Finally, we evaluate DDL on two different Amazon[1] datasets and show its consistent superiority to the competing baselines.

The main contributions of this paper are summarized as follows:

(1) We propose a hashing-based hybrid recommendation framework by combining hashing and hybrid recommendation techniques, which effectively overcome sparse and cold-start issues, and significantly speed up online recommendation.
(2) We integrate a deep learning model into our hashing based recommendation framework in a unified way, which is helpful to extract efficient binary representations. Hence DDL provides a good trade-off between recommendation efficiency and accuracy.
(3) We add balance and irrelevant constraints on hash codes and develop an alternating optimization algorithm to solve the proposed discrete mixed-integer programming problem, which is helpful to extract compact and informative hash codes and can improve the recommendation performance.

The rest of this paper is organized as follows: Section 2 and Section 3 introduce the related works and the preliminary, respectively. Section 4 introduces the model proposed in this paper in detail. Section 5 introduces the initialization and the discrete optimization algorithm to solve the proposed model. Section 6 introduces the experimental settings for DDL and other comparison methods. Section 7 analyzes the experimental results and Section 8 concludes the paper.

2 RELATED WORK

In this section, we review several major schemes closely relevant to this paper. Firstly, we introduce several real-valued hybrid recommender systems proposed to alleviate data sparsity by combining user-item interaction and content data. We then present the latest two types of hashing-based recommendation frameworks.

2.1 Real-Valued Hybrid Recommendation

2.1.1 CTR. Collaborative topic regression [23] is a state-of-the-art hybrid recommender system, which was proposed by combining topic model, collaborative filtering, and probabilistic matrix factorization (PMF) [17]. The authors developed a machine learning algorithm for recommending scientific articles to users in an online scientific community. CTR obtained real latent representations of users and items by exploiting two types of data: user's collection data and article content data. It can be used to mitigate cold start and data sparsity settings.

2.1.2 CDL. Collaborative deep learning [24] was proposed as a probabilistic model by jointly learning a probabilistic stacked denoising autoencoder (SDAE) [21] and CF. Similar to CTR, CDL exploits interaction and content data to alleviate cold start and data sparsity problems. Differ from CTR, CDL took advantage of deep learning framework to learn effective real latent representations. Thus it can be applied in cold-start and sparse settings. CDL is a tightly coupled method for recommender systems by developing a hierarchical Bayesian model.

2.1.3 CKE. Collaborative knowledge base embedding [34] is an integrated framework composed of three components: heterogeneous network embedding method, stacked denoising auto-encoders, and stacked convolutional auto-encoders. CKE extracted items semantic representations from structural content, textual content and visual content, and it was proposed to deal with cold start and data sparsity settings.

2.1.4 VBPR. Visual Bayesian Personalized Ranking [7] is a factorization model by incorporating visual features into predictors of users' preferences. By utilizing visual features

[1] https://www.amazon.com/

extracted from product images by (*pre-trained*) deep networks, VBPR is also helpful to alleviate cold start and sparse issues.

2.2 Hashing-based Recommendation

Below we mainly review recent advances of hashing-based recommendation frameworks. For comprehensive review of hashing techniques, please refer to [26].

2.2.1 Two-stage Hashing Recommendation. As discussed in Section 1, the two-stage framework consists of relaxed optimization stage and binary quantization stage. Real latent representations can be obtained by the relaxed optimization, and hash codes can be obtained by the quantization. A pioneer work [3] was proposed to exploit Locality-Sensitive Hashing [4] to generate hash codes for Google News readers based on their click history. On this basis, A. Karatzoglou et al. [10] randomly projected real latent representations learned from regularized matrix factorization into hash codes. Similar to this, K. Zhou et al.[39] followed the idea of Iterative Quantization [5] to generate binary codes from rotated real latent representations. To derive compact binary codes, the uncorrelated constraints were imposed on the real latent representations in regularized matrix factorization. However, according to the analysis in [38], hashing essentially only preserves similarity rather than preference based on inner product, since the magnitudes of the representations for users and items are discarded in the quantization stage. Thus, a constant feature norm (CFN) constraint was imposed when learning the real latent representations, and then the magnitudes and similarity are respectively quantized in [37, 38]. But the two-stage approach still suffered large information loss in the quantization procedure.

2.2.2 Hashing Learning Recommendation. Differ from two-stage hashing frameworks, hashing learning frameworks can obtain hash codes by directly solving the discrete optimization problem. Thus more information is carried by hash codes than two-stage frameworks. Zhang et al. proposed discrete collaborative filtering (DCF) [35], which is a hashing learning recommendation framework. By adding balance and uncorrelated constraints on hash codes, DCF obtained efficient binary codes. DCF was evaluated using a similar way of the conventional CF [16]. By directly optimizing a ranking evaluation metric – the Area Under ROC (Receiver Operating Characteristics) Curve (AUC), discrete personalized ranking (DPR) [36] was proposed to learn hash codes under the same constraints with DCF, and it also obtained short and informative hash codes. Since the above hashing learning frameworks are based on CF, thus they still suffer low recommendation accuracy under sparse setting, and they cannot work when new users or items present.

3 PRELIMINARY AND PROBLEM STATEMENT

In this section, we first introduce some notations used in this paper and then formulate the problem. After that, we briefly introduce the deep learning framework, DBN, used in this paper.

Table 1: Notations

Symbol	Size	Description
$\mathbf{b}_i$	$r \times 1$	hash code of user i
$\mathbf{d}_j$	$r \times 1$	hash code of user j
V	$\mathcal{S}$	the index set of observed ratings
V_i	$\mathcal{S}_i$	items set rated by user i
V_j	$\mathcal{S}_j$	users set of item j
$\mathbf{c}_j$	$P \times 1$	bag-of-words of item j
Θ	1×2	parameter of DBN
$\mathbf{W}$	$1 \times l$	weight matrices of l-layer DBN
$\mathbf{b}$	$1 \times l$	bias vectors of l-layer DBN
$\mathbf{B}$	$r \times n$	hash codes matrix of all users in I
$\mathbf{D}$	$r \times m$	hash codes matrix of all items in J
$\mathbf{X}$	$r \times n$	delegated real matrix of $\mathbf{B}$
$\mathbf{Y}$	$r \times m$	delegated real matrix of $\mathbf{D}$

3.1 Notations

Assume that there are n users and m items in a recommender system. The users set is defined as I and the items set is indicated as J. The observed rating $s_{ij} \in S$ represents the preference of user i for item j. The items content data is denoted as $\mathbf{C}$, which consists of bag-of-words vectors of all items in J. The other essential notations used in this paper are listed in Table 1.

3.2 Problem Statement

The goal of recommender system is to provide items that can match users' preferences. Preference model is formulated based on the representations of users and items, which is dependent on the recommendation framework.

Cold-Start: Cold-start problem consists of cold-start user and cold-start item. Specifically, cold-start item problem is caused by new items which have no interaction information [18]. Conducting recommendation under the two cold-start settings is a challenge for recommender systems, and hence it often suffers low accuracy.

Data Sparsity: As introduced in [14], data sparsity refers to a situation where interaction data is insufficient for identifying similar users/items, and it is a major factor that affects the performance of recommendation.

Problem Definition(Discrete Deep Learning): Given a user-item rating dataset S, a content dataset $\mathbf{C}$, and a query user i, our goal is to recommend top-k items that user i would be interested in. The problem becomes a cold-start item recommendation if the predicted top-k items are not in the rating dataset S, and meanwhile in the content data C.

3.3 Deep Belief Network

To obtain effective item representations, we use DBN to extract deep hierarchical item representations from item content data. DBN [8] is a generative probabilistic model consists of one input layer and multiple hidden layers. Jointly training all layers is computationally intractable. Hinton et al. put forward an efficient algorithm to train DBN in a greedy layerwise manner [8], in which the hidden layers are trained once at a time in a bottom-up way, and the above unsupervised procedure is called *pre-train*. After adding an extra learning objective, the learned representation by *pre-train* stage is converted into a supervised learning process,

then *fine-tune* all the parameters of the DBN with respect to concerning the additional learning objective (in this paper, the objective in Section 5.4). The *fine-tune* process is usually implemented by Back-Propagation (BP) algorithm [9].

4 DISCRETE DEEP LEARNING

In this section, we first model users' preferences by hash codes, then we introduce the proposed DDL.

4.1 Preference Model

The preference model is a key component to formulate a recommender system. The traditional model is usually based on Matrix Factorization [12] under a low-rank assumption. Users and items are projected into a r-dimension real latent space, and the preference of user i for item j is estimated by the inner product of real-valued vectors. Similarly, the proposed hashing-based framework in this paper maps users and items into a r-dimension Hamming space, in which the preference is estimated by the Hamming distance between hash codes of users and items.

Suppose that hash codes of user i and item j are denoted by $\mathbf{b}_i \in \{\pm 1\}^r$ and $\mathbf{d}_j \in \{\pm 1\}^r$, respectively. The preference of user i for item j is defined as

$$\begin{aligned}\widehat{p}_{ij} &= 1 - \frac{1}{r}\sum_{k=1}^{r}\mathbb{I}\left(b_{ik} \neq d_{jk}\right) \\ &= \frac{1}{r}\sum_{k=1}^{r}\mathbb{I}\left(b_{ik} = d_{jk}\right) \\ &= \frac{1}{2} + \frac{1}{2r}\mathbf{b}_i^T\mathbf{d}_j \end{aligned} \tag{1}$$

where $\mathbb{I}(\cdot)$ is an indicator function that returns 1 if the input is true, otherwise it returns 0. $\widehat{p}_{ij}$ is in the range of 0 to 1 and that represents the predicted preference of user i over item j. From the above preference model, it can be observed that the preference model in r-dimension Hamming space is consistent with that in r-dimension real latent space.

4.2 Discrete Deep Learning

Discrete Deep Learning (DDL) is a hashing-based hybrid recommendation framework which adds hashing technique into the hybrid recommender framwork, that consists of DBN and CF by exploiting rating and item content data. Hence, DDL can obtain effective hash codes by jointly optimizing two objectives: DBN based objective and CF based objective.

For each item j, the real-valued representation $\mathbf{f}_j$ can be automatically learnt by the *pre-train* procedure of DBN from the bag-of-words $\mathbf{c}_j$:

$$\mathbf{f}_j = \mathrm{DBN}(\mathbf{c}_j, \Theta).$$

To address the issues of data sparsity and cold start item, our DDL effectively exploits the content information of items using the deep learning model DBN. Thus, one of our objectives is to minimize the difference between the learnt representation $\mathbf{f}_j$ and the binary representation (hash code) $\mathbf{d}_j$ for each item j, and it is given by

$$\arg\min_{\mathbf{D},\ \Theta} \sum_{j=1}^{m} \|\mathbf{d}_j - \mathrm{DBN}(\mathbf{c}_j, \Theta)\|_F^2, \tag{2}$$

where $\Theta = \{\mathbf{W}, \mathbf{b}\}$ is the parameters of a l-layer DBN that contains weight matrices $\mathbf{W} = \{\mathbf{W}^{(1)}, \mathbf{W}^{(2)}, \cdots, \mathbf{W}^{(l)}\}$ and bias vectors $\mathbf{b} = \{\mathbf{b}^{(1)}, \mathbf{b}^{(2)}, \cdots, \mathbf{b}^{(l)}\}$, $\mathbf{D}$ is stacked by $\mathbf{d}_j$, where $j \in J$. Therefore, by minimizing the objective function in Equation (2), we obtain effective items' hash codes by the deep hierarchical framework from content data.

The CF based objective is to minimize the difference between the observed rating s_{ij} and the predicted preference $\widehat{p}_{ij}$. We assume that $s_{ij} \in \mathbf{S}$ is in the range of 0 to 1 denoting the rating of user i to item j, and get the following objective from Equation (1):

$$\arg\min_{\mathbf{B},\mathbf{D}} \sum_{(i,j)\in V} \left(s_{ij} - \frac{1}{2} - \frac{1}{2r}\mathbf{b}_i^T\mathbf{d}_j\right)^2, \tag{3}$$

where V is the index set of S, $\mathbf{B}, \mathbf{D}$ respectively are the hash codes of users and items, and r is the hash code length. By minimizing the above objective, we can obtain effective hash codes of users and items from the observed rating data.

By combining the above two objectives of Equation (2) and Equation (3), the objective function of the proposed DDL is

$$\begin{aligned}\arg\min_{\mathbf{B},\mathbf{D},\boldsymbol{\Theta}} & \sum_{(i,j)\in V}\left(2rs_{ij} - r - \mathbf{b}_i^T\mathbf{d}_j\right)^2 \\ & + \frac{\lambda}{2}\sum_{j=1}^{m}\|\mathbf{d}_j - \mathrm{DBN}(\mathbf{c}_j, \Theta)\|_F^2 \\ s.t.\ \ & \mathbf{B} \in \{\pm 1\}^{r\times n}, \mathbf{D} \in \{\pm 1\}^{r\times m}, \end{aligned} \tag{4}$$

where $\lambda > 0$ is a tuning parameter that weights the importance of the two objectives. In order to maximize the entropy of each binary bit, it is needed to add a balance constraint, so that each bit carries as much information as possible [39]. In addition, to learn compact binary codes, irrelevant constraints also need to be imposed, that guarantees each bit is independent of others. In other words, there is no redundant information in the obtained hash codes. After adding the above two additional constraints on both $\mathbf{B}$ and $\mathbf{D}$, we can reformulate the problem Equation (4) as

$$\begin{aligned}\arg\min_{\mathbf{B},\mathbf{D},\Theta} & \sum_{(i,j)\in V}\left(2rs_{ij} - r - \mathbf{b}_i^T\mathbf{d}_j\right)^2 \\ & + \frac{\lambda}{2}\sum_{j=1}^{m}\|\mathbf{d}_j - \mathrm{DBN}(\mathbf{c}_j, \Theta)\|_F^2 \\ s.t.\ \ & \mathbf{B} \in \{\pm 1\}^{r\times n}, \mathbf{D} \in \{\pm 1\}^{r\times m}, \\ & \mathbf{B}\mathbf{1}_n = \mathbf{0}, \mathbf{D}\mathbf{1}_m = \mathbf{0}, \mathbf{B}\mathbf{B}^T = n\mathbf{I}_r, \mathbf{D}\mathbf{D}^T = m\mathbf{I}_r, \end{aligned} \tag{5}$$

where $\mathbf{1}_n$ ($\mathbf{1}_m$) present n-dimension (m-dimension) vectors that all elements are 1, and $\mathbf{I}_r$ is a $r \times r$-dimension identity matrix. As Equation (5) is a discrete optimization problem, which is proven to be an intractable NP-hard problem [6], we adopt a methodology like [35] to soften the balance and irrelevant constraints. Specifically, we add delegated real valued matrices $\mathbf{X}$ and $\mathbf{Y}$ to approximate hash codes $\mathbf{B}$ and $\mathbf{D}$, respectively. Thus Equation (5) can be rewritten as:

$$\begin{aligned}\arg\min_{\mathbf{B},\mathbf{D},\Theta} & \sum_{(i,j)\in V}\left(2rs_{ij}-r-\mathbf{b}_i^T\mathbf{d}_j\right)^2+\alpha\left\|\mathbf{B}-\mathbf{X}\right\|_F^2 \\ & +\frac{\lambda}{2}\sum_{j=1}^{m}\left\|\mathbf{d}_j-\mathrm{DBN}(\mathbf{c}_j,\Theta)\right\|_F^2+\beta\left\|\mathbf{D}-\mathbf{Y}\right\|_F^2, \\ s.t.\quad & \mathbf{B}\in\{\pm1\}^{r\times n},\mathbf{D}\in\{\pm1\}^{r\times m} \\ & \mathbf{X}\mathbf{1}_n=\mathbf{0},\mathbf{Y}\mathbf{1}_m=\mathbf{0},\mathbf{X}\mathbf{X}^T=n\mathbf{I}_r,\mathbf{Y}\mathbf{Y}^T=m\mathbf{I}_r, \quad (6)\end{aligned}$$

where α and β are tuning parameters so that the second and last terms in Equation (6) allow certain discrepancy between $\mathbf{B}$ and $\mathbf{X}$, and between $\mathbf{D}$ and $\mathbf{Y}$. Since $tr(\mathbf{B}\mathbf{B}^T)=tr(\mathbf{X}\mathbf{X}^T)=nr$ and $tr(\mathbf{D}\mathbf{D}^T)=tr(\mathbf{Y}\mathbf{Y}^T)=mr$ are constant. Thus the objective in Equation (6) can be equivalently transformed as the following mixed integer optimization problem:

$$\begin{aligned}\arg\min_{\mathbf{B},\mathbf{D},\Theta,\mathbf{X},\mathbf{Y}} & \sum_{(i,j)\in V}\left(a_{ij}-\mathbf{b}_i^T\mathbf{d}_j\right)^2-2\alpha tr(\mathbf{B}^T\mathbf{X}) \\ & +\frac{\lambda}{2}\sum_{j=1}^{m}\left\|\mathbf{d}_j-\mathrm{DBN}(\mathbf{c}_j,\Theta)\right\|_F^2-2\beta tr(\mathbf{D}^T\mathbf{Y}) \\ s.t.\quad & \mathbf{B}\in\{\pm1\}^{r\times n},\mathbf{D}\in\{\pm1\}^{r\times m} \\ & \mathbf{X}\mathbf{1}_n=\mathbf{0},\mathbf{Y}\mathbf{1}_m=\mathbf{0},\mathbf{X}\mathbf{X}^T=n\mathbf{I}_r,\mathbf{Y}\mathbf{Y}^T=m\mathbf{I}_r, \quad (7)\end{aligned}$$

where $a_{ij}=2rs_{ij}-r$. Differ from two-stage hashing-based frameworks, we do not discard the binary constraints through the objective transformations.

4.3 Advantage of DDL for Online Recommendation

4.3.1 Time Complexity. **Real-valued space:** As discussed in Section 1, suppose there are n users and m items in a recommender system, after extracting real latent representations, predicting top-k items in real space has the time complexity of $\mathcal{O}(nmr+nm\log k)$. **Hamming space:** After hash codes have been obtained, we can make a recommendation by ranking the predicted preferences by Equation (1). For each user (denoted by hash code), ranking items by the predicted preferences is equivalent to finding the top-k nearest items (denoted by hash codes) in Hamming space. As discussed in [38], searching nearest neighbors in Hamming space is extremely fast. There are two methods to search the top-k items: one is Hamming ranking, it can be conducted by ranking Hamming distances with the query hash code (user), it has the complexity of $\mathcal{O}(m)$, which is linear with the item data size. The other one is hashing lookup. Similar hash codes are searched in a hamming ball centered at the query hash code. The time complexity is independent of the item data size. Therefore, Hashing based recommendation has evident superiority over the real-valued recommendation.

4.3.2 Storage complexity. **Real space:** At least 64 bits are needed to store a double real number. When the dimension of real latent vectors becomes large, it will cost much more space. **Hamming space:** Only one bit is needed to store a binary code. Thus the storage cost is reduced significantly. If we store m items by r-dimension real latent vectors, it will cost $64mr$ bits, while it costs mr bits by r-dimension hash codes. Moreover, if we store hash codes with sparse vectors, the storage cost will be reduced further [1, 38].

5 MODEL OPTIMIZATION

In this section, an alternating optimization strategy is developed to solve the mixed integer optimization problem shown in Equation (7). We first initialize all parameters of DDL in Section 5.1. We then introduce how to update $\mathbf{B}$, $\mathbf{D}$, Θ, $\mathbf{X}$, and $\mathbf{Y}$, respectively.

5.1 Initialization

We initialize DDL by the *pre-train* procedure of DBN. Specifically, we first learn the item representations $\mathbf{f}_j$ by the *pre-train* process of DBN. Then we initialize Θ as the result of the *pre-train*, and initialize each $\mathbf{d}_j$ as the sign of $\mathbf{f}_j$. Besides, we randomly initialize $\mathbf{X}$ and $\mathbf{Y}$ by the standard normal distribution and initialize $\mathbf{B}$ as the sign of $\mathbf{X}$.

5.2 Update B given D, X, Y, and Θ

Since the objective function in Equation (7) sums over users independently, we update $\mathbf{B}$ by updating $\mathbf{b}_i$ in parallel by minimizing the following objective function:

$$\arg\min_{\mathbf{b}_i\in\{\pm1\}^r}\sum_{j\in V_i}\left(\mathbf{d}_j^T\mathbf{b}_i\right)^2-2\sum_{j\in V_i}a_{ij}\mathbf{d}_j^T\mathbf{b}_i-2\alpha\mathbf{x}_i^T\mathbf{b}_i, \quad (8)$$

where V_i is the items set rated by user i. This discrete optimization problem is NP-hard, and thus we adopt a bitwise learning strategy named Discrete Coordinate Descent (DCD) [19] to update $\mathbf{b}_i$. Particularly, let b_{ik} be the k-th bit of $\mathbf{b}_i$ and let $\mathbf{b}_{i\bar{k}}$ be the rest bits of $\mathbf{b}_i$, i.e, $\mathbf{b}_i=[\mathbf{b}_{i\bar{k}}^T,b_{ik}]^T$. Note that DCD can update b_{ik} given $\mathbf{b}_{i\bar{k}}$. Discarding the terms independent of b_{ik}, the objective function in Equation (8) is rewritten as

$$\arg\min_{b_{ik}\in\{\pm1\}}\hat{b}_{ik}b_{ik}, \quad (9)$$

where $\hat{b}_{ik}=\sum_{j\in V_i}\mathbf{d}_{j\bar{k}}^T\mathbf{b}_{i\bar{k}}d_{jk}-\sum_{j\in V_i}a_{ij}d_{jk}-\alpha x_{ik}$. Due to the space limit, we omit the derivation details. The above objective function reaches the minimal only if b_{ik} had the opposite sign of $\hat{b}_{ik}$. However, if $\hat{b}_{ik}$ was zero, b_{ik} would not be updated. Therefore, the update rule of b_{ik} is

$$b_{ik}=sgn\left(K\left(-\hat{b}_{ik},b_{ik}\right)\right), \quad (10)$$

where $K(t,r)=t$ if $t\neq0$, otherwise, $K(t,r)=r$.

5.3 Update D given B, X, Y, and Θ

Provided $\mathbf{B}$, $\mathbf{X}$, $\mathbf{Y}$, and Θ are fixed, discarding terms irrelevant to $\mathbf{d}_j$ in Equation (7), we formulate the following subproblem as:

$$\begin{aligned}\arg\min_{\mathbf{d}_j\in\{\pm1\}^r} & \sum_{i\in V_j}\left(\mathbf{b}_i^T\mathbf{d}_j\right)^2-2\sum_{i\in V_j}a_{ij}\mathbf{b}_i^T\mathbf{d}_j \\ & +\frac{\lambda}{2}\|\mathbf{d}_j-\mathrm{DBN}(\mathbf{c}_j,\Theta)\|_F^2-2\beta\mathbf{y}_j^T\mathbf{d}_j, \quad (11)\end{aligned}$$

where V_j is the users set that rated item j. Similarly, we optimize $\mathbf{d}_j$ by the bitwise learning, and the objective is rewritten as

$$\arg\min_{b_{ik}\in\{\pm1\}}\hat{d}_{jk}d_{jk}, \quad (12)$$

where $\hat{d}_{jk} = \sum_{i \in V_j} \mathbf{b}_{i\bar{k}}^T \mathbf{d}_{j\bar{k}} b_{ik} - \sum_{i \in V_j} a_{ij} b_{ik} - \frac{\lambda}{2} \text{DBN}(\mathbf{v}_j, \Theta)_k + \beta y_{jk}$. As discussed in Section 5.3, the update rule of d_{jk} is:

$$d_{jk} = sgn\left(K\left(-\hat{d}_{jk}, d_{jk}\right)\right). \quad (13)$$

5.4 Update Θ given B, D, X, and Y

If $\mathbf{B}$, $\mathbf{D}$, $\mathbf{X}$, and $\mathbf{Y}$ are fixed, the optimization problem Equation (7) can be rewritten as

$$\arg\min_{\Theta} \sum_{j=1}^{m} \|\mathbf{d}_j - \text{DBN}(\mathbf{c}_j, \Theta)\|^2. \quad (14)$$

Since $\mathbf{D}$ is fixed, Equation (14) is a supervised DBN learning framework. As introduced in Section 3.3, after initialization, all parameters need to be *fine-tuned* by using stochastic gradient descent method, where the gradient descent part is implemented by BP algorithm. As $\mathbf{d}_j \in \{\pm 1\}^r$, we choose *tanh* function as the output function of DBN since its output is in the range of -1 to 1 that has the same range with hash codes. We choose sigmoid function as the activation function for hidden layers.

5.5 Update X given B, D, Y, and Θ

Given $\mathbf{B}$, $\mathbf{D}$, $\mathbf{Y}$, and Θ, the Equation (7) is transformed as

$$\underset{\mathbf{X} \in \mathbb{R}^{r \times n}}{\text{argmax}}\ tr(\mathbf{B}^T \mathbf{X}),\ s.t.\ \mathbf{X}\mathbf{1}_n = \mathbf{0}, \mathbf{X}\mathbf{X}^T = n\mathbf{I}_r. \quad (15)$$

It can be solved with the help of SVD according to [35, 36]. Specifically, $\mathbf{X}$ is updated by

$$\mathbf{X} = \sqrt{n}[\mathbf{U}_s\ \widehat{\mathbf{U}}_s][\mathbf{V}_s\ \widehat{\mathbf{V}}_s]^T, \quad (16)$$

where $\mathbf{U}_s$ and $\mathbf{V}_s$ are respectively stacked by the left and right singular vectors of the row-centered matrix $\bar{\mathbf{B}} : \bar{b}_{ij} = b_{ij} - \frac{1}{n}\sum_{i=1}^{n} b_{ij}$. $\widehat{\mathbf{U}}_s$ is stacked by the left singular vectors and $\widehat{\mathbf{V}}_s$ can be calculated by Gram-Schmidt orthogonalization, and it satisfies $[\mathbf{V}_s\ \mathbf{1}]^T \widehat{\mathbf{V}}_s = \mathbf{0}$.

5.6 Update Y with fixed B, D, X, and Θ

Given $\mathbf{B}$, $\mathbf{D}$, $\mathbf{X}$, and Θ, the Equation (7) can be transformed as

$$\underset{\mathbf{Y} \in \mathbb{R}^{r \times m}}{\text{argmax}}\ tr(\mathbf{D}^T \mathbf{Y}),\ s.t.\ \mathbf{Y}\mathbf{1}_\mathbf{m} = \mathbf{0}, \mathbf{Y}\mathbf{Y}^T = m\mathbf{I}_r. \quad (17)$$

Similar to Section 5.5, $\mathbf{Y}$ can be updated by

$$\mathbf{Y} = \sqrt{m}[\mathbf{P}_s\ \widehat{\mathbf{P}}_s][\mathbf{Q}_s\ \widehat{\mathbf{Q}}_s]^T. \quad (18)$$

Similarly, $\mathbf{P}_s$, $\mathbf{Q}_s$, $\widehat{\mathbf{P}}_s$, and $\widehat{\mathbf{Q}}_s$ can be determined by the row-centered matrix of $\mathbf{D}$.

6 EXPERIMENTAL SETTINGS

In this section, we first introduce datasets used in our experiments. Then we introduce two evaluation methods and five state-of-the-art comparison methods.

6.1 Datasets

We adopt the Amazon dataset[2] in our experiments, which is one of the biggest, most comprehensive, and publicly available datasets for the study of recommendation. It covers user interactions (such as ratings) on items as well as item content (such as item metadata and descriptions) on 24 product categories spanning May 1996 - July 2014, and each group contains a sub-dataset. We adopt two of the largest product categories Clothing, Shoes & Jewelry, and Cell Phones & Accessories for experiments, and the two datasets are briefly denoted as 'Cloth' and 'Cell' in the following. We use rating and item content data in our experiments. The rating matrix has a sparsity of 99.9%. Some statistics of the datasets are shown in Table 2.

Table 2: Statistics of datasets.

Dataset	#User(n)	#Item(m)	#Rating($\mathcal{S}$)	Sparsity(%)
'Cloth'	39,387	23,033	278,653	99.97%
'Cell'	27,879	10,429	194,340	99.93%

6.1.1 Data Pre-processing. For rating data, we normalize each rating into the interval of [0,1] to keep consistent with the predicted preference defined in this paper. For content data, we first remove punctuations, numbers, stop words, and words with the length smaller than two since these words usually have no discriminative meanings, we then conduct stemming on the remaining words by the Porter Stemmer [15]. Finally, similar to [23], by ranking the TF-IDF values we choose the top 8000 discriminative words from the two datasets to form dictionaries separately, then we get the bag-of-words $\mathbf{C}$ for all items.

6.1.2 Data Spliting. To simulate sparse settings, similar to [13], we take different proportions (10%, 20%) of ratings as training set D_{train} and the remaining positive ratings (i.e., original ratings $\geq$ 4 stars) D_{test} are used for testing. Taking 20% as an example, we randomly select 20% ratings for each item as the training data, and the remaining positive ratings are chosen for testing. The random selection is carried out 5 times independently, and we report the experimental results as the average values.

6.2 Evaluation Methods

As introduced in Section 3.2, the goal of recommendation is to find out the top-k items that users may be interested in. We adopt two common ranking evaluation methods: Accuracy@k and Mean Reciprocal Rank (MRR), to evaluate the quality of the ranking list. Accuracy@k was widely adopted by many previous ranking based recommender systems [2, 11], and MRR was also widely used as a metric for ranking tasks [20, 22].

The basic idea of Accuracy@k is to test whether a user's favorite item appears in the predicted top-k items list, we define the user's *favorite (positive) items* as the ones rated 4-stars or 5-stars (0.8 or 1 rating in this paper), and the *positive ratings* refer to ratings 0.8 or 1. For each positive rating $s_{ij} \in D_{test}$: (1) we randomly choose 1000 negative items and compute predicted rating scores for the ground-truth item j as well as the 1000 negative items; (2) we form a ranked list by ordering these items according to their predicted ratings; (3) if the ground-truth item j appears in the top-k ranked list, we have a hit; otherwise, we have a miss.

6.2.1 Accuracy@k. Accuracy@k has been widely used in evaluating recommendation accuracy by assessing the quality

[2] http://jmcauley.ucsd.edu/data/amazon/

of the obtained top-k items list. Accuracy@k is formulated as:

$$\text{Accuracy@}k = \frac{\#hit@k}{|D_{test}|},$$

where $|D_{test}|$ is the size of the test set, and $\#hit@k$ denotes the number of hits in the test set.

6.2.2 MRR. The Mean Reciprocal Rank (MRR) [22] is to evaluate a ranking task that produces a list of responses to a query, ordered by probability of correctness. The reciprocal rank of a query response is the multiplicative inverse of the rank of the first correct answer. The mean reciprocal rank is the average of the reciprocal ranks of results for a query, MRR is defined as:

$$\text{MRR} = \frac{1}{|D_{test}|} \sum\nolimits_{s_{ij} \in D_{test}} \frac{1}{rank(i,j)},$$

where $rank(i,j)$ is the position of item j in the obtained top-k items list for user i.

6.3 Comparison Methods and Settings

As introduced in Section 2.2, we choose two types of comparison methods that consist of three real-valued hybrid recommender systems (CTR, CDL, and VBPR) and two latest competing hashing-based recommender systems (DCF and DPR).

In our experiments, we use 5-fold cross validation method on randomly splits of training data, to tune the optimal hyper-parameters for our algorithm and all the compared algorithms: CTR, CDL, VBPR, DCF, and DPR. We perform grid search to find the optimal hyper-parameters.

For CTR, we set $\lambda_u = 0.1$, $\lambda_v = 10$, $a = 1$, $b = 0.01$ and $K = 50$ since it can achieve good performance. For CDL, it can achieve good performance when we set $\lambda_u = 1$, $\lambda_v = 10$, $\lambda_n = 1e^4$, $\lambda_w = 1e^{-4}$, $a = 1$ and $b = 0.01$. For aligning with the dimension of other methods, we set $K = 30$, and the layer structure of SDAE is set as $[8000, 200, 30]$. For VBPR, we set $\lambda_\Theta = 10$.

For DCF, we search α and β from $\{1e^{-4}, 1e^{-3}, \cdots, 1e^{2}\}$, and the optimal parameters are $\alpha = 1e^{-3}$ and $\beta = 1e^{-3}$. For DPR, we find the optimal parameters respectively are $\alpha = 1e^{-4}$ and $\beta = 1e^{-3}$.

For DDL proposed in this paper, we search α, β from $\{1e^{-4}, 1e^{-3}, \cdots, 1e^{2}\}$ and λ from $\{1e^{-1}, 1, \cdots, 1e^{2}\}$ by grid search. As a result, we set the layer structure of DBN as $[8000, 800, 30]$, and set $\alpha = 1e^{-3}$, $\beta = 1e^{-3}$ and $\lambda = 10$.

7 EXPERIMENTAL RESULTS

In this section, we first display the efficiency of hashing-based recommendation compared with real-valued hybrid recommendation frameworks on 'Cloth' and 'Cell' datasets. Then, we verify the effectiveness of our proposed DDL from three aspects: Firstly, we evaluate the accuracy in recommending cold-start items; Secondly, we explore the accuracy effectiveness in two sparse settings; Thirdly, we present the accuracy effectiveness compared with the competing hashing baselines.

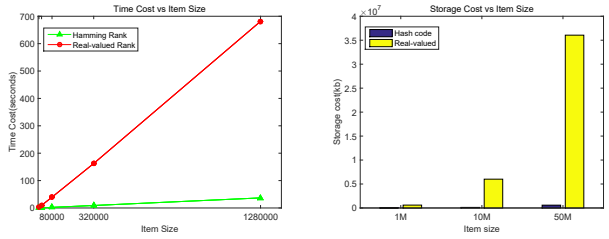

Figure 1: Efficiency comparison on artificial data. Left: Time cost comparison. Right: Storage cost comparison

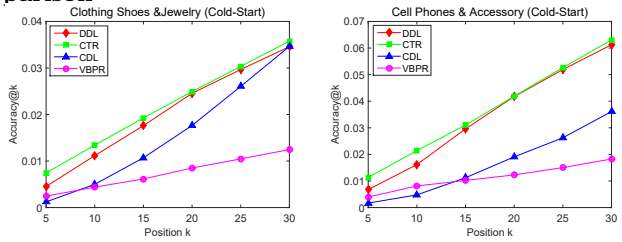

Figure 2: Accuracy@k in cold-start item setting

7.1 Efficiency Comparison

As analyzed in Section 4.3, the significant advantage of hashing recommendation over the real-valued recommendation is the efficiency of online recommendation. We separately evaluate the efficiency in terms of time and storage on synthetic data.

7.1.1 Time Complexity. We investigate the time cost of preference ranking when the item number varies. We use standard gaussian distribution to generate items' real-valued features randomly. Items hash codes are obtained from real-valued vectors by the sign function. We set different sizes of items sets in the experiment: 5000, 20000, 80000, 320000, 128000, to test the time cost variation of online recommendation. The variation is shown in the left of Figure 1. We conclude that the time cost of real-valued features grows fast with item number, in comparison, the time cost of hash codes increase much slower than real-valued features. The experimental results show that hashing based recommendation has evident advantage over the real-valued recommendation for online recommendation.

7.1.2 Storage Complexity. We test the storage costs of hash codes and real-valued features on 3 different sizes of item sets: 1 million, 10 million, and 50 million. From the right of Figure 1, hash codes cost much less memory to store the same number of items, which is consistent with the analysis in Section 4.3.

7.2 Accuracy Comparison

7.2.1 Accuracy on Cold-Start Item Recommendation. This experiment studies the accuracy comparison between the existing real-valued hybrid recommender systems and DDL under the same cold-start item setting. We test the performance on the D_{test} (10%) introduced in Section6.1. Specifically, we first choose items with less than 5 positive ratings as cold-start items and then select users with at least one positive rating as test users. For each test user, we first choose his/her ratings related to cold-start items as the test set, and

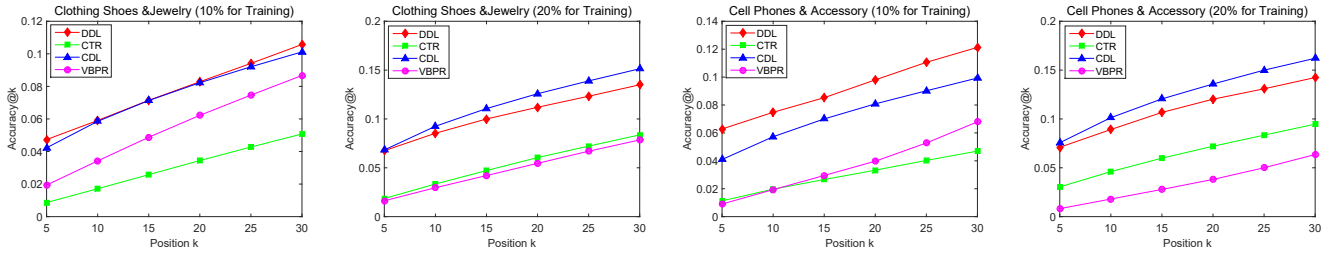

Figure 3: Comparison with Real-valued methods on 'Cloth' and 'Cell' datasets w.r.t two sparse settings.

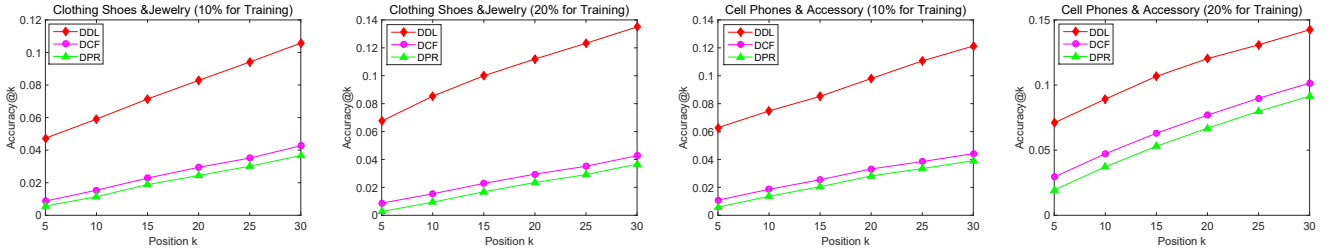

Figure 4: Comparison with Hashing methods on 'Cloth' and 'Cell' datasets w.r.t two sparse settings.

Table 3: MRR in cold-start item setting

Method	CTR	CDL	VBPR	DDL
'Cloth'	0.0082*	0.0071	0.0042	0.0077^o
'Cell'	0.0117*	0.0072	0.0053	0.0099^o

the remaining ratings as the training set. Our goal is to test whether the marked-off cold-start items can be accurately recommended to the right user.

We adopt Accuracy@k and MRR to respectively evaluate recommendation accuracy. The accuracy@k comparison of two Amazon datasets is shown in Figure 2. Our proposed DDL performs almost as well as the best result of the real-valued hybrid recommender systems. Table 3 summarizes MRR results for the four algorithms, the best result is marked as '$\star$' and the second best is marked as 'o' . We can find that the performance of DDL is very close to the best result, that is consistent with the outcome of Accuracy@k.

The real-valued hybrid recommender systems implement a recommendation by real-valued features, while DDL produces a recommendation by hash codes. As evaluated in Section 7.1, DDL has superiority in the efficiency of online recommendation over real-valued hybrid recommender systems. Due to real latent vectors intuitively carried more information than hash codes. Thus it is acceptable and reasonable to have small gaps between real-valued hybrid recommendation and hashing based DDL.

7.2.2 Accuracy on Sparse Recommendation. Our task in this section is to assess the accuracy effectiveness of DDL compared with real-valued hybrid recommendation on two sparse settings (10%, 20%) introduced in Section 6.1. The results of Accuracy@k and MRR are respectively presented in Figure 3 and Table 4. We can find that DDL performs best when the rating data is very sparse (10%). With the observed ratings becoming dense (20%), CDL performs best under Accuracy@k, and DDL performs best under MRR. In a word, the two metrics have consistent performance in sparse settings. The results indicate that DDL provides a good trade-off between efficiency and accuracy, and DDL can also make effective recommendation by hash codes in sparse settings.

Table 4: MRR on 'Cloth' and 'Cell' datasets with two spare settings (10%, 20%)

	'Cloth'		'Cell'	
	10%	20%	10%	20%
CTR	0.0102	0.0168	0.0113	0.0245
CDL	0.0311^o	0.0481^o	0.0300^o	0.0526^o
VBPR	0.0176	0.0158	0.0130	0.0123
DDL	0.0490*	0.0669*	0.0584*	0.0605*

7.2.3 Accuracy Compared with Hashing Frameworks. In this section, we evaluate the effectiveness of DDL compared with two competing hashing baselines. Figure 4 shows the results of Accuracy@k on two datasets in two sparse settings, respectively. We see that DDL outperforms DCF and DPR. DCF and DPR can not work very well in sparse settings since they are based on CF.

8 CONCLUSION

In this paper, a content-aware hashing approach called Discrete Deep Learning (DDL) is proposed to alleviate data sparsity and cold-start item problem. First, we formulate a preference model based on hash codes. Second, based on the preference model, we present the DDL recommendation framework by adding a binary constraint on the combination objective of the supervised DBN and CF-based objectives. By imposing uncorrelated and independent constraints on hash codes, compact and informative hash codes are directly learned by an alternating optimization method. Third, we evaluate the effectiveness of DDL regarding Accuracy@k and MRR on two Amazon datasets. Experiments show that DDL has an obvious advantage over the competing baselines in cold-start item and sparse settings. DDL provides a good trade-off between recommendation efficiency and accuracy.

9 ACKNOWLEDGEMENT

This work was supported by ARC Discovery Early Career Researcher Award (Grant No. DE160100308), ARC Discovery Project (Grant No. DP170103954) and New Staff Research Grant of The University of Queensland (Grant No.613134). It was also supported by National Natural Science Foundation of China (Grant No. 61572335, 61572109, 61502077, 61631005) and the Fundamental Research Funds for the Central Universities (Grant No. ZYGX2016J087).

REFERENCES

[1] Wenlin Chen, James Wilson, Stephen Tyree, Kilian Weinberger, and Yixin Chen. 2015. Compressing neural networks with the hashing trick. In *International Conference on Machine Learning*. 2285–2294.

[2] Wen-Yen Chen, Jon-Chyuan Chu, Junyi Luan, Hongjie Bai, Yi Wang, and Edward Y Chang. 2009. Collaborative filtering for orkut communities: discovery of user latent behavior. In *Proceedings of the 18th international conference on World wide web*. ACM, 681–690.

[3] Abhinandan S Das, Mayur Datar, Ashutosh Garg, and Shyam Rajaram. 2007. Google news personalization: scalable online collaborative filtering. In *Proc. of WWW*. ACM, 271–280.

[4] Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S Mirrokni. 2004. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the twentieth annual symposium on Computational geometry*. ACM, 253–262.

[5] Yunchao Gong, Svetlana Lazebnik, Albert Gordo, and Florent Perronnin. 2013. Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35, 12 (2013), 2916–2929.

[6] Johan Håstad. 2001. Some optimal inapproximability results. *J. ACM* 48, 4 (2001), 798–859.

[7] Ruining He and Julian McAuley. 2016. VBPR: visual bayesian personalized ranking from implicit feedback. In *Thirtieth AAAI Conference on Artificial Intelligence*.

[8] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. 2006. A fast learning algorithm for deep belief nets. *Neural computation* 18, 7 (2006), 1527–1554.

[9] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. 1989. Multilayer feedforward networks are universal approximators. *Neural networks* 2, 5 (1989), 359–366.

[10] Alexandros Karatzoglou, Markus Weimer, and Alex J Smola. 2010. Collaborative filtering on a budget. In *International Conference on Artificial Intelligence and Statistics*. 389–396.

[11] Yehuda Koren. 2008. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 426–434.

[12] Yehuda Koren and Robert Bell. 2011. Advances in collaborative filtering. In *Recommender systems handbook*. Springer, 145–186.

[13] Hao Ma, Haixuan Yang, Michael R Lyu, and Irwin King. 2008. Sorec: social recommendation using probabilistic matrix factorization. In *Proceedings of the 17th ACM conference on Information and knowledge management*. ACM, 931–940.

[14] Manos Papagelis, Dimitris Plexousakis, and Themistoklis Kutsuras. 2005. Alleviating the sparsity problem of collaborative filtering using trust inferences. In *International Conference on Trust Management*. Springer, 224–239.

[15] Martin F Porter. 1980. An algorithm for suffix stripping. *Program* 14, 3 (1980), 130–137.

[16] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence*. AUAI Press, 452–461.

[17] Ruslan Salakhutdinov and Andriy Mnih. 2007. Probabilistic Matrix Factorization.. In *Nips*, Vol. 1. 2–1.

[18] Andrew I Schein, Alexandrin Popescul, Lyle H Ungar, and David M Pennock. 2002. Methods and metrics for cold-start recommendations. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 253–260.

[19] Fumin Shen, Chunhua Shen, Wei Liu, and Heng Tao Shen. 2015. Supervised Discrete Hashing. In *CVPR*. 37–45.

[20] Yue Shi, Alexandros Karatzoglou, Linas Baltrunas, Martha Larson, Nuria Oliver, and Alan Hanjalic. 2012. CLiMF: learning to maximize reciprocal rank with collaborative less-is-more filtering. In *Proceedings of the sixth ACM conference on Recommender systems*. ACM, 139–146.

[21] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. 2010. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research* 11, Dec (2010), 3371–3408.

[22] Ellen M Voorhees et al. 1999. The TREC-8 Question Answering Track Report. In *Trec*, Vol. 99. 77–82.

[23] 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.

[24] Hao Wang, Naiyan Wang, and Dit-Yan Yeung. 2015. Collaborative deep learning for recommender systems. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1235–1244.

[25] Jun Wang, Sanjiv Kumar, and Shih-Fu Chang. 2012. Semi-supervised hashing for large-scale search. *IEEE TPAMI* 34, 12 (2012), 2393–2406.

[26] Jun Wang, Wei Liu, Sanjiv Kumar, and Shih-Fu Chang. 2016. Learning to hash for indexing big dataA survey. *Proc. of the IEEE* 104, 1 (2016), 34–57.

[27] Weiqing Wang, Hongzhi Yin, Shazia Sadiq, Ling Chen, Min Xie, and Xiaofang Zhou. 2016. Spore: A sequential personalized spatial item recommender system. In *Data Engineering (ICDE), 2016 IEEE 32nd International Conference on*. IEEE, 954–965.

[28] Hongzhi Yin, Bin Cui, Ling Chen, Zhiting Hu, and Xiaofang Zhou. 2015. Dynamic user modeling in social media systems. *ACM Transactions on Information Systems (TOIS)* 33, 3 (2015), 10.

[29] Hongzhi Yin, Bin Cui, Yizhou Sun, Zhiting Hu, and Ling Chen. 2014. Lcars: A spatial item recommender system. *ACM Transactions on Information Systems (TOIS)* 32, 3 (2014), 11.

[30] Hongzhi Yin, Bin Cui, Xiaofang Zhou, Weiqing Wang, Zi Huang, and Shazia Sadiq. 2016. Joint modeling of user check-in behaviors for real-time point-of-interest recommendation. *ACM Transactions on Information Systems (TOIS)* 35, 2 (2016), 11.

[31] Hongzhi Yin, Yizhou Sun, Bin Cui, Zhiting Hu, and Ling Chen. 2013. Lcars: a location-content-aware recommender system. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 221–229.

[32] Hongzhi Yin, Weiqing Wang, Hao Wang, Ling Chen, and Xiaofang Zhou. 2017. Spatial-Aware Hierarchical Collaborative Deep Learning for POI Recommendation. *IEEE Transactions on Knowledge and Data Engineering* 29, 11 (2017), 2537–2551.

[33] Hongzhi Yin, Xiaofang Zhou, Bin Cui, Hao Wang, Kai Zheng, and Quoc Viet Hung Nguyen. 2016. Adapting to user interest drift for poi recommendation. *IEEE Transactions on Knowledge and Data Engineering* 28, 10 (2016), 2566–2581.

[34] Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. 2016. Collaborative knowledge base embedding for recommender systems. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 353–362.

[35] Hanwang Zhang, Fumin Shen, Wei Liu, Xiangnan He, Huanbo Luan, and Tat-Seng Chua. 2016. Discrete collaborative filtering. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*. ACM, 325–334.

[36] Yan Zhang, Defu Lian, and Guowu Yang. 2017. Discrete Personalized Ranking for Fast Collaborative Filtering from Implicit Feedback. (2017).

[37] Yan Zhang, Guowu Yang, Lin Hu, Hong Wen, and Jinsong Wu. 2017. Dot-product based preference preserved hashing for fast collaborative filtering. In *Communications (ICC), 2017 IEEE International Conference on*. IEEE, 1–6.

[38] Zhiwei Zhang, Qifan Wang, Lingyun Ruan, and Luo Si. 2014. Preference preserving hashing for efficient recommendation. In *Proc. of SIGIR*. ACM, 183–192.

[39] Ke Zhou and Hongyuan Zha. 2012. Learning binary codes for collaborative filtering. In *Proc. of ACM SIGKDD*. ACM, 498–506.